Open source en países en desarrollo: Condiciones para un apoyo oficial aumentador de bienestar

Scalise, Luciano

2009

Universidad de Buenos Aires
Facultad de Ciencias Económicas
Maestría en Economía

Open Source en Países en Desarrollo. Condiciones Para un Apoyo Oficial Aumentador de Bienestar.

Alumno: *Luciano Scalise*

Tutor: *Enrique Chaparro*

ABSTRACT

Las recomendaciones de política pública en relación al Free Libre Open Source Software (FLOSS), son sensibles respecto al supuesto que se realice sobre la existencia de externalidades de red; planteándose así la necesidad de investigar argumentos teóricos y empíricos que permitan determinar la pertinencia de suponer dichas externalidades de red para el sector del software en países en desarrollo, de forma tal que la intervención gubernamental a favor del open source incremente el bienestar agregado de la sociedad. Esto constituye el problema de investigación de la presente tesis, centrando el análisis empírico en datos correspondientes a la Argentina (país en desarrollo), versus las series correspondientes al total mundial. El aporte logrado mediante dicha comparación, se extendería mas allá del presente trabajo, brindando conocimiento para futuras investigaciones, ya que permitiría entender hasta que punto es posible extrapolar los resultados y conclusiones cualitativas dependientes de la existencia de externalidades de red en la industria del software, obtenidas para el mercado global (dominado por países desarrollados), a los mercados en desarrollo. Desde el punto de vista metodológico, el presente trabajo brindaría una vía alternativa a las usualmente utilizadas, para el análisis empírico de externalidades de red.

Índice.

1. INTRODUCCIÓN.

La difusión del modelo de producción de software FLOSS (Free Libre Open Source) podría tener efectos positivos de magnitud significativa sobre el desarrollo económico, especialmente para países menos industrializados. Por este motivo, cobra importancia evaluar la viabilidad de las distintas formas de intervención gubernamental en apoyo del movimiento FLOSS en términos de sus efectos (¿beneficios?) potenciales para los distintos agentes económicos y para la sociedad en su conjunto, particularmente en países menos desarrollados.

Para ello, tenemos que entender no solamente que motiva el nacimiento de programadores sino cómo se difunden los productos por ellos creados. De hecho, **las recomendaciones de política pública en relación al FLOSS, son sensibles respecto al supuesto que se realice sobre la existencia de externalidades de red; planteándose así la necesidad de investigar argumentos teóricos y empíricos que permitan determinar la pertinencia de suponer dichas externalidades de red para el sector del software en países en desarrollo, de forma tal que la intervención gubernamental a favor del open source incremente el bienestar agregado de la sociedad. Esto constituye el problema de investigación de la presente tesis, centrando el análisis empírico en datos correspondientes a la Argentina (país en desarrollo), versus las series correspondientes al total mundial. El aporte logrado mediante dicha comparación, se extendería mas allá del presente trabajo, brindando conocimiento para futuras investigaciones, ya que permitiría entender hasta que punto es posible extrapolar los resultados y conclusiones cualitativas dependientes de la existencia de externalidades de red en la industria del software, obtenidas para el mercado global (dominado por países desarrollados), a los mercados en desarrollo.**

1.1. Externalidades de red. Conceptos generales.

Las externalidades de red pueden definirse como aquellos efectos que hacen que "el valor de un producto o servicio para un usuario dependa no sólo del producto en sí mismo sino del número de usuarios que utilicen dicho producto o servicio" (Fuentelsaz et al, 2002).

Aunque se ha realizado una considerable cantidad de investigación teórica sobre las externalidades de red, el número de trabajos empíricos resulta mucho más limitado, debido principalmente a la escasez de datos, tal y como señalan numerosos investigadores (Gowrisankaran y Stavins, 2004 / Stavins, 2003 / Kauffman, McAndrews y Wang, 2000 / Brynjolfsson y Kemerer, 1996), y la mayoría de estos trabajos empíricos se han centrado en una única categoría de productos. El camino seguido en el presente trabajo en cuanto a contrastaciones empíricas, también se basará en determinadas categorías de productos, por los motivos antes referidos.

La clave para la aparición de externalidades de red es la existencia de cierta complementariedad y/o interacción entre la tecnología de los distintos usuarios individuales (Economices, 1996 / Economides y Himmelberg, 1995), lo que produce dos efectos fundamentales en la dinámica de la Industria (Katz y Shapiro, 1986):

- Se modifica el atractivo de la red, generando economías de escala del lado de la demanda. Esto implica que el precio que los usuarios pagan está en parte determinado por el tamaño de la red a la que pertenece el producto (Brynjolfsson y Kemerer, 1996).
- Los potenciales consumidores considerarán en su decisión de compra las expectativas futuras de éxito de las distintas redes en competencia.

En el ámbito de las TIC (***T***echnology and ***I***nformation ***C***ompanies) es frecuente que aparezca este tipo de externalidades (Rohlfs, 1974) debido a que algunas de las redes de tecnología presentan muchas similitudes con las redes reales. Como afirman Shapiro y Varian (1999: 165) "Hay una diferencia fundamental entre la "nueva" y la "antigua" economía: la vieja economía industrial estaba impulsada por las economías de escala; la nueva economía de la información está impulsada por la economía de las redes". Sin embargo, aunque resultan muy frecuentes, no siempre aparece este tipo de efectos, y pese a que su presencia resulta obvia en determinados ámbitos, en ocasiones es preciso recurrir a un estudio detallado para detectar su existencia.

Los efectos de red producen una dinámica en los mercados que hace a los fuertes más fuertes (círculo virtuoso) y los débiles más débiles (círculo vicioso), de modo que en muchos casos se produce la adopción de una única tecnología, quedando el resto eliminadas, fenómeno conocido como winer takes all (McGee y Sammut, 2002). Pero esto no implica que la competencia sea escasa, sino todo lo contrario: la competencia hasta que una compañía logre establecer su tecnología como dominante puede ser muy intensa (Economides, 2001). De hecho una de las peculiaridades de los mercados en que aparecen este tipo de efectos es que no resulta extraño que una determinada tecnología sea superada por otra técnicamente inferior, como por ejemplo en el caso de los vídeos VHS y Betamax (McGee y Sammut, 2002).

La clave de esta adopción ineficiente es la dependencia de las condiciones iniciales, que es una consecuencia del proceso de realimentación positiva: pequeñas diferencias en las cuotas de mercado iniciales pueden suponer una gran diferencia en la evolución del mercado (Schilling, 2002 y 1998 / Wade, 1995 / Arthur, 1989 y 1990).

En relación con esta idea de *path dependence* (para un análisis detallado de los distintos tipos de path dependence ver Liebowitz y Margolis, 1995) surge el concepto de *Lock-in*, que aparece cuando un usuario

o grupo de usuarios quedan vinculados a una tecnología debido a que los costos de cambio son elevados (Amit y Zott, 2001 / Shapiro y Varian, 1999: 111-124), de modo que se refuerza la posición de una tecnología ya instalada como consecuencia de los costos, tanto individuales como sociales, que implica la adopción de un nuevo estándar tecnológico.

Sin embargo, incluso existiendo externalidades de red, la heterogeneidad en las preferencias de los consumidores (su valoración de las distintas tecnologías), así como la diferenciación de productos pueden hacer que varias redes coexistan de forma simultánea, ya que determinados usuarios pueden preferir la ventajas intrínsecas de un producto aunque esto implique pertenecer a una red de menor tamaño. El caso de las computadoras Apple es un buen ejemplo de este tipo de situaciones (Van Hove, 1999).

Existen numerosos ejemplos de competencia entre tecnologías sujetas a efectos de red, y entre ellos podemos citar la competencia entre corriente alterna y continua en el siglo XIX , la elección del ancho de vía para el ferrocarril, los teclados de ordenador QWERTY y DVORAK, el vídeo VHS frente al Betamax y en la actualidad el VHS frente al DVD, los CDs frente a las cintas de audio, la competencia entre hojas de cálculo, los sistemas operativos para PCs y un largo etcétera (ver por ejemplo Schilling, 2002 / Shapiro y Varian, 1999: 197-213 / Loch y Huberman, 1999 / Gates, 1998: 49-54 / David, 1984).

Los beneficios externos generados por un usuario al conectarse a una red pueden estar causados por varios factores, dando lugar a tres tipos de externalidades (ver por ejemplo Zodrow, 2003 / Amit y Zott, 2001 / Goolsbee y Zittrain, 1999 / Keilbach y Posch, 1998 / Yoffie, 1996 / Katz y Shapiro, 1985):

- **Directas.** Se producen cuando el valor de conectarse a una red se incrementa con el número de puntos de comunicación, por lo que la clave es precisamente el incremento de la capacidad de comunicarse con otros usuarios. Por ejemplo, la utilidad para un consumidor de adquirir un teléfono depende del número de teléfonos ya instalados con los que pueda establecer comunicación. Análogamente emplear un procesador de texto ampliamente extendido garantiza que cualquier documento elaborado con el mismo podrá ser leído por un gran número de individuos.
- **Indirectas.** Debidas a los mecanismos estándar de mercado. Al incrementarse el número de usuarios de una red se producirá una bajada de precios en los productos (economías de escala), al tiempo que se incrementará la variedad de productos complementarios y su facilidad de compra, con lo que los potenciales clientes se verán beneficiados. Para un comprador de hardware, por ejemplo, el número de otros compradores de un hardware similar resulta un factor importante porque la cantidad y variedad de software compatible será una función creciente con el número de usuarios. El trabajo de Liebowitz y Margolis (1994) probó que existe dicha relación entre la decisión de compra de los usuarios y la variedad y precio de los productos complementarios. No obstante, en la industria del software, esto no será necesariamente así, ya que al consolidarse una posición dominante en el mercado, combinada

con inelasticidades de demanda originadas en efectos *lock – in*; podrían incentivar incrementos de precio una vez se conseguida la masa crítica.

- **Aprendizaje.** Al aumentar el tamaño de la red se incrementará el número de usuarios con conocimientos específicos sobre la tecnología asociada. Estos "expertos", poniendo a disposición de otros usuarios sus conocimientos, favorecen la expansión de la red, de modo que un usuario logrará un mejor servicio post venta además del consejo de otros usuarios experimentados. Por otra parte, quienes ya conocen la tecnología, incurrirían en un coste de aprendizaje en caso de querer adoptar otra distinta. Existen numerosos ejemplos de este efecto en la literatura, como el caso de las máquinas de escribir QWERTY discutido por David (1984) o el trabajo empírico de Goolsbee y Klenow (2002) sobre la difusión de ordenadores en Estados Unidos, que demostró que existía una probabilidad de compra del primer ordenador mucho mayor en ciudades con un elevado número de usuarios de ordenadores, hecho que atribuyeron a las Externalidades de aprendizaje.

Una idea clave relacionada con la curva de demanda es el concepto de masa crítica de usuarios, que puede definirse como el tamaño mínimo de la red para que a los potenciales usuarios les compense incorporarse a la misma (Rohlfs, 1974 / Oren y Smith, 1982 / Oren, Smith y Wilson, 1982), es decir, el tamaño mínimo requerido para iniciar la realimentación positiva. Para tamaños inferiores a la masa crítica los potenciales usuarios no están interesados en incorporarse a la red, e incluso los usuarios que ya se han incorporado tenderán a abandonarla, de modo que la tecnología puede fracasar. Una vez superado este punto, la red se irá expandiendo hasta alcanzar su tamaño de equilibrio.

1.2. Marco actual.

Durante los pasados 5 años, ha resurgido el interés académico en el desarrollo de software Open Source (de código abierto). Una de las encrucijadas académicas es entender qué es lo que motivó a los programadores a desarrollador este tipo de producto. Se ha argumentado, que la actividad de programadores open sources se vio alentada por tres factores:

- *Externalidades de red:* Un número importante de productos Open Source (OSS), tales como Apache Web Server, dominan sus categorías de producto. En el mercado de sistemas operativos para computadoras personales, International Data Corporation estima que el sistema OSS Linux tiene entre siete y veintiún millones de usuarios, con una tasa de crecimiento del 200% anual. Muchos observadores creen que representa un desafío potencial de magnitud para el sistema Windows de Microsoft en este segmento importante del mercado.
- *La significativa inversión de capital en proyectos OSS.* Durante los pasados cinco años, numerosas grandes corporaciones, incluyendo a Hewlett Packard, IBM, y Sun, han lanzado proyectos para desarrollar y utilizar OSS (***O***pen ***S***ource ***S***oftware). Mientras tanto, un numero de compañías

especializadas en la comercialización de Linux, tales como Red Hat y VA Linux, han completado ofertas publicas iniciales, y otras compañías de OSS tales como Cobalt Networks (posteriormente adquirida por Sun Microsystems), Collab.Net, Scriptics, y Sendmail han recibido financiación a través de venture capitals.

- *La nueva estructura organizacional.* La naturaleza cooperativa del desarrollo OSS, han sido aclamada en el negocio y prensa técnica como una importante innovación organizacional que atrae la participación de más y más programadores.

Constituye un desafío intelectual para el economista representativo, explicar el comportamiento de programadores individuales comprometidos en el proceso OSS. Cabe referir a las dos citas siguientes:

> La idea que el sistema social de software propietario – el sistema que dice que no esta permitido compartir o cambiar software – es antisocial, es decir antitético, simplemente equivocado, puede sorprender a alguna gente. ¿Pero que más se podría decir de un sistema basado en dividir al público y mantener indefensos a los usuarios? [Stallman, 1999].

> La "función de utilidad" que maximiza un hacker de Linux no es económicamente clásica, si no que tiene explicación a través del intangible que representa la satisfacción de su propio ego y reputación entre otros hackers. […..]. Culturas voluntarias que funcionan de este modo no son inusuales; otro ámbito similar en el cual he participado durante mucho tiempo es la ciencia ficción, en la cual opuestamente al hackeo, explícitamente se reconoce el "egoboom" (el enriquecimiento de la reputación personal entre otros fans). [Raymond, 1999].

En principio no estaría claro como estas cuestiones se relacionan con la visión tradicional del proceso de innovación en la literatura económica. ¿Por que miles de programadores de elite contribuyen gratis ha la provisión de un bien publico? Cualquier explicación basada en altruismo no avanza mucho. Mientras que usuarios en países menos desarrollados sin duda se benefician del acceso al software gratis, muchos otros beneficiarios son individuos adineraros o incluso compañías del ranking Fortune 500. Más aun, el altruismo no ha jugado un rol importante en otras industrias, de manera que tendría que ser explicado porque los individuos en la industria del software son más altruistas que en otras industrias.

1.3. Definiciones y contexto histórico.

Software libre (en inglés free software) es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente, sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. Análogamente, el software gratis o gratuito (denominado usualmente freeware) incluye en algunas ocasiones el código fuente; aunque este tipo de software, no es libre en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

No debe confundirse "software libre" con software de dominio público. Éste último es aquél por el que no es necesario solicitar ninguna licencia y cuyos derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquél cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es dominio público. En resumen, el software de dominio público es la pura definición de la libertad de usufructo de una propiedad intelectual que tiene la humanidad porque así lo ha decidido su autor o la ley, tras un plazo contado desde la muerte de éste, habitualmente 70 años.

El termino free, traducido al castellano, significa tanto libre como gratis, por eso muchas veces suelen confundirse el freeware con el software libre aunque entre ambos existen notables diferencias.

De acuerdo con tal definición, el software es "libre" si garantiza las siguientes libertades:

- Libertad 0, ejecutar el programa con cualquier propósito (privado, educativo, público, comercial, militar, etc.).
- Libertad 1, estudiar y modificar el programa (para lo cual es necesario poder acceder al código fuente).
- Libertad 2, copiar el programa de manera que se pueda ayudar al vecino o a cualquiera.
- Libertad 3, Mejorar el programa y publicar las mejoras.

Es importante señalar que las libertades 1 y 3 obligan a que se tenga acceso al código fuente. La "libertad 2" hace referencia a la libertad de modificar y redistribuir el software libremente licenciado bajo algún tipo de licencia de software libre que beneficie a la comunidad.

El término software no libre, se emplea para referirse al software distribuido bajo una licencia de software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del copyright; el software dispuesto bajo una licencia de software libre rescinde específicamente la mayoría de estos derechos reservados.

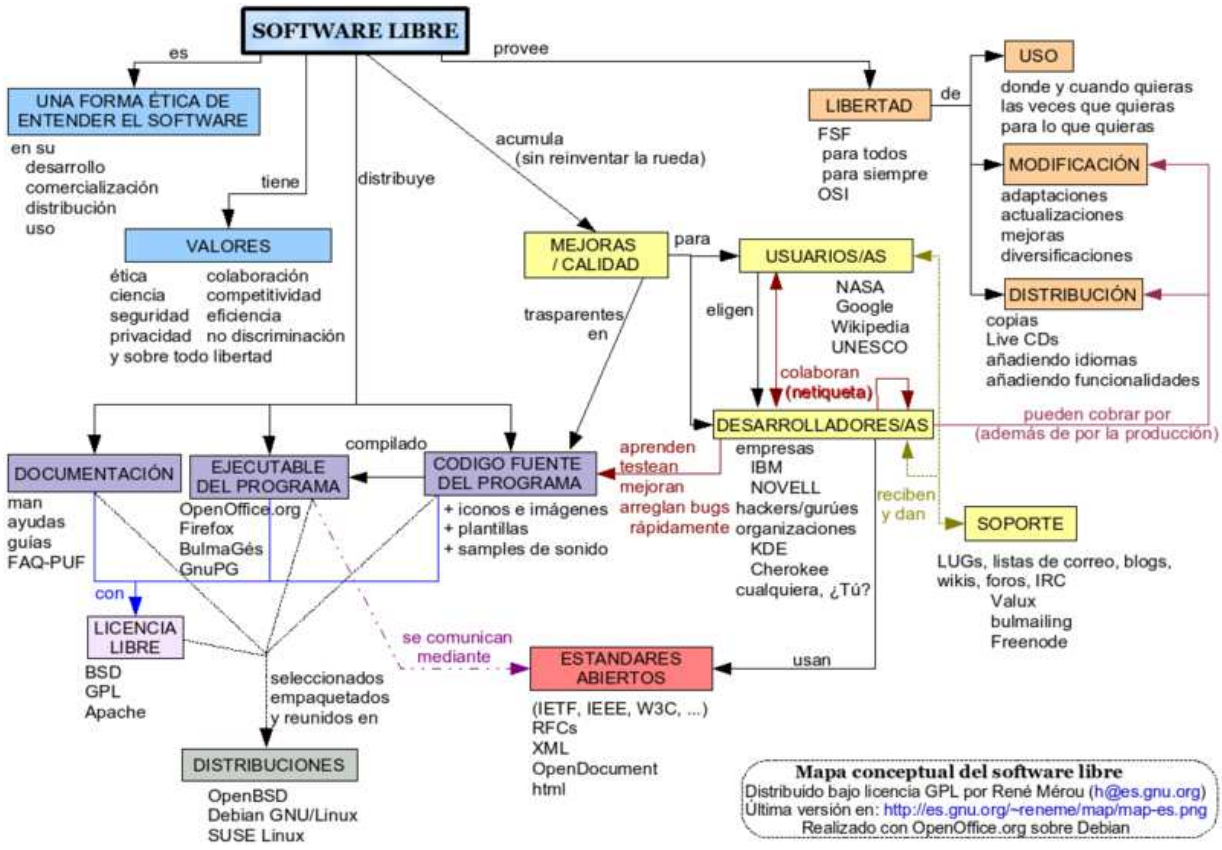
La definición de software libre no contempla el asunto del precio; un eslogan frecuentemente usado es "libre como en libertad, no como en cerveza gratis" o en inglés "Free as in freedom, not as in free beer" (aludiendo a la ambigüedad del término inglés "free"), y es habitual ver a la venta CDs de software libre como distribuciones Linux. Sin embargo, en esta situación, el comprador del CD tiene el derecho de copiarlo y redistribuirlo. El software gratis pude incluir restricciones que no se adaptan a la definición de software libre - por ejemplo, puede no incluir el código fuente, puede prohibir explícitamente a los distribuidores recibir una compensación a cambio, etc.

Para evitar la confusión, algunas personas utilizan los términos "libre" (Libre software) y "gratis" (Gratis software) para evitar la ambigüedad de la palabra inglesa "free". Sin embargo, estos términos alternativos son usados únicamente dentro del movimiento del software libre, aunque están extendiéndose lentamente hacia el resto del mundo. Otros defienden el uso del término Open Source Software (software de código abierto, también llamado de fuentes abiertas). La principal diferencia entre los términos "Open Source" y "Free Software" es que éste último tiene en cuenta los aspectos éticos y filosóficos de la libertad, mientras que el "Open Source" se basa únicamente en los aspectos técnicos.

En un intento por aunar los mencionados términos que se refieren a conceptos semejantes, se está extendiendo el uso de la palabra "FLOSS" con el significado de "Free - Libre - Open Source Software" e, indirectamente, también a la comunidad que lo produce y apoya.

Una licencia es aquella autorización formal con carácter contractual que un autor de un software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del software libre, existen distintas variantes del concepto o grupos de licencias.

La mayoría de la gente lo interpreta como un esquema de distribución, e intercambia libremente "open source" con "software libre". Aun cuando existen importantes diferencias filosóficas entre ambos términos, especialmente en términos de las motivaciones para el desarrollo y el uso de tal software, raramente suelen tener impacto en el proceso de colaboración. Por tal motivo, a lo largo de presente trabajo, se tomará la libertad de intercambiar estos términos.



El presente trabajo posee la siguiente estructura: en el capitulo 2 se presentará el marco teórico para el análisis de las implicancias del FLOSS sobre el desarrollo económico, se estudiarán las diferencias entre el modelo de producción del software propietario y el FLOSS, y serán descritas las principales características de las políticas públicas mas usuales sobre el Free Libre Open Source Software. El capitulo 3 estará dedicado a detallar y aplicar la metodología propuesta para la contratación de la hipótesis relevante, poniendo especial énfasis en la teoría econométrica de modelos VAR, dado que la misma conforma el núcleo central del trabajo empírico desarrollado. En el capitulo 4 se presentarán los resultados obtenidos utilizando la metodología del capitulo precedente, extrayéndose las conclusiones e implicancias derivadas a partir de dichos resultados.

2. ESTADO DE LA CUESTIÓN.

De acuerdo a lo planteado por Varian y Shapiro (Varian y Shapiro, 2003), la primer obligación de los oficiales públicos realizando elecciones acerca de plataformas de software es elegir el sistema más apto para la tarea a realizar. Existen muchos casos en los que la performance, confiabilidad, y seguridad de los sistemas de código abierto es igual o superior a la de las alternativas propietarias. Cuando dos sistemas poseen similar aptitud para una tarea, las alternativas de open source se vuelven importantes ya que típicamente reducen los costos de interconexión y de switching, haciendo menos factible que el consumidor se vea atrapado por un solo vendedor (efecto lock – in). Las interfases abiertas alientan a los desarrolladores de terceras partes a crear sus propias aplicaciones, adds – on, y productos complementarios. Los beneficios de tales productos para los usuarios, y aun para todo el país, son tan grandes, que virtualmente todos los vendedores quieren *proclamar apertura.* Pero la pregunta importante es si realmente tienen incentivos para llevar a cabo esta proclama, no solo en el presente, si no también a lo largo del camino cuando los costos de switching a un sistema alternativo sean considerablemente altos.

Existirían virtualmente tres opciones en el mercado del software:

- Software propietario.
- Software propietario copiado ilegalmente.
- Software FLOSS (Free Libre Open Source Software).

La opción correspondiente al software propietario copiado ilegalmente, actúa como "segunda marca" del mismo, creando base instalada adicional (existiría un trade – off entre ganancias por ventas y creación de base instalada por esta vía, pero cabe señalar que los productos mas copiados ilegamente, serían los que representan mayor utilidad para las empresas productoras de software propietario). Esta situación, produciría interferencia en la libre competencia de mercado entre el software propietario y FLOSS.

La mayoría de los usuarios, solo podrían considerar como viables dos de las tres opciones antes referidas, debido a distintos tipos de restricciones. Así, por ejemplo, ciertos usuario consideran las opciones de software propietario legal o ilegal, pero no el FLOSS, debido a falta de información; mientras que las corporaciones, debido a las penalidades que le podrían ser inflingidas, solo considerarían el software legalmente adquirido, sea propietario o FLOSS. Esta observación respecto a las opciones viables para el ámbito empresarial, podría explicar la mayor adopción de FLOSS registrada para este tipo de usuarios.

Países deseosos de estimular una industria domestica del software fuerte deberían mirar primero a su sistema universitario y pensar "¿qué entorno será mas útil para educar a nuestros futuros desarrolladores de software?". Las interfases abiertas son críticas ya que permiten el desarrollo local de aplicaciones de

terceras partes. Plataformas open source y plataformas comerciales con interfases abiertas, ambas ofrecen oportunidades para las compañías de software locales. En contraste, las plataformas comerciales con interfases propietarias, pueden dejar a los desarrolladores de terceras partes en una desventaja estratégica en relación a la compañía en control de las interfases entre la plataforma y las aplicaciones. Tal dependencia estratégica puede desalentar la inversión local en términos de dinero y recursos humanos en el desarrollo de software de aplicación comercial.

El OSS también juega un rol importante en tanto expone el funcionamiento interno del software a los estudiantes de modo que estos puedan observar como se ensambla software de calidad. Tal como aspirantes a mecánico de automóviles necesitan trabajar sobre motores reales, aspirantes a ingenieros de sistemas necesitan trabajar sobre sistemas operativos reales. Tener tales sistemas disponibles, y abiertos al escrutinio, llevara a tener mejores computadores científicos, y mejores productos en el futuro.

2.1. Principales diferencias entre el modelo de software propietario y el FLOSS.

En esta sección, se seguirá en lo esencial, lo expuesto por Lerner y Tirole (Lerner y Tirole, 2000).

Algunos miembros de la comunidad informática, creen que este método de producción (FLOSS) domina al de desarrollo tradicional de software en todos los aspectos. Pero muchos adherentes al FLOSS discuten que el software open source tiende a estar equipado para usuarios mas sofisticados. Este punto es expuesto de manera colorida por un desarrollador de software open source:

> En cada ciclo de lanzamiento Microsoft siempre escucha a sus *clientes más ignorantes*. Esta es la clave para restarle inteligencia a cada lanzamiento del software y lograr mayor penetración entre la población que se encuentra fuera del ambiente de usuarios de computadoras personales. Los desarrolladores de Linux y OS/2, por otro lado, tienden a escuchar a sus *consumidores mas inteligentes*.....El bien que Microsoft hace al acercar a las computadoras a los no – usuarios, lo deshace por la carga que impone a los usuarios experimentados. [Nadeau, 1999].

Ciertamente, la mayor difusión de los proyectos open source parece darse en entornos en los cuales el usuario final es sofisticado, tales como Apache Server el cual es instalado por administradores de sistemas. En estos casos, los usuario parecerían están mas dispuestos a tolerar la falta de documentación o de interfases fáciles – de – entender, a cambio del ahorro en los costos y la posibilidad de modificar los códigos fuente ellos mismos.

A continuación, se analizarán algunos de los aspectos más importantes en la comparación entre el modelo de desarrollo de software propietario y el modelo open source.

2.1.1. Incentivos generales en el modelo de software propietario vs. Open Source.

Los proyectos comerciales tienen una ventaja en la dimensión de la compensación corriente, porque la naturaleza propietaria del código genera ingresos. Esto hace que valga la pena para las empresas privadas ofrecer sueldos. Este argumento económico clásico según el cual las perspectivas de ganancia alientan la inversión, es utilizado, por ejemplo, para justificar el otorgamiento de patentes para promover la inversión.

En contraste, un proyecto de FLOSS encarado por una empresa privada, puede reducir el costo de la mano de obra (programadores) por dos razones:

2.1.1.1. *Efecto alumno:* dado que el código es distribuido gratuitamente, puede ser utilizado en escuelas y universidades para propósitos educativos; así sería ya familiar al programador (mayor disponibilidad de mano de obra). Esto reduciría el costo de programar en UNIX, por ejemplo.

2.1.1.2. Customización y solución de defectos: el costo de contribuir a un proyecto de open source es menor si la actividad trae aparejado un beneficio privado (solución de defectos, customización) para el programador a nivel de su firma. Debe notarse que este factor de reducción de costo se relaciona directamente con la apertura del código fuente.

2.1.2. Incentivos de señalización para los programadores:

Los incentivos de señalización pueden ser mayores en el modelo open source por tres razones:

2.1.2.1 *Mejor medición de performance:* los observadores externos solo pueden ver de manera inexacta la funcionalidad y/o calidad de los elementos individuales de un típico programa desarrollado comercialmente, dado que no pueden acceder al código fuente de características propietarias. En contraste, en un proyecto open source (código abierto), el observador externo puede acceder no solo a ver cual fue la contribución de cada individuo y si esa contribución funcionó, si no que también puede ver si la tarea fue difícil, si el problema fue resuelto de manera inteligente, si el código puede ser útil a otros programadores en el futuro, etc.

2.1.2.2 Iniciativa total: el programador open source es su propio jefe y toma responsabilidad total por el éxito de su sub – proyecto. En una firma comercial

jerárquica, sin embargo, la performance del programador depende de la interferencia y consejos de su supervisor. La teoría económica sugeriría que la performance del programador sería mejor medida en el primer caso.

2.1.2.3 Mayor fluidez: podría discutirse que el mercado laboral es mas fluido en un entorno open source. Los programadores son más propensos a tener menos capital ideosincratico o especifico a la firma, que limite el cambio hacia un nuevo programa o entorno de trabajo. (Dado que muchos elementos del código fuente son compartidos a través de proyectos open source, más del conocimiento que han acumulado pude ser transferido al nuevo entorno).

De acuerdo a lo expuesto en esta sección, existen argumentos suficientes para pensar que el movimiento OSS posee incentivos para continuar atrayendo programadores que brinden sustento a la "base creativa", y al mismo tiempo, las empresas podrían encontrar atractivas algunas de las características de este modelo de producción de software. De este modo, el movimiento OSS podría pensarse como una alternativa de largo plazo, y no solo como una "moda" del mercado del software.

2.2. Políticas publicas sobre el Free Libre Open Source Software (FLOSS).

En esta sección, se analizaran los meritos e inconveniente de las distintas formas de intervención gubernamental directa o indirecta sobre los proyectos FLOSS.

2.2.1. Modelización sin discriminación entre usuarios informados y no informados.

El mercado del software difiere de los mercados estándar o tradicionales, en al menos tres aspectos importantes que pueden dar lugar a fallas de mercado: (i) grandes economías de escala, (ii) innovaciones crucialmente importantes, (iii) significativas externalidades de red y costos de switching.

El esquema teórico desarrollado por Schmidt y Schnitzer (Schmidt y Schnitzer, 2002), no realiza discriminación entre usuarios informados y no informados, cobrando especial relevancia los aspectos antes señalados, a la hora de estudiar las distintas alternativas de intervención gubernamental. A continuación, se analizan dichas alternativas bajo el marco teórico de estos autores.

2.2.1.1. Subsidios directos para proyectos Open Source específicos.

Esta alternativa de intervención gubernamental existe potencialmente, pero habría consenso (incluso entre la literatura de distintas corrientes) respecto a que reduciría el bienestar social conjunto (Schmidt y Schnitzer, 2002 / Comino y Manenti, 2003).

El gobierno podría apoyar en forma directa proyectos OSS particulares a través de la oferta de subsidios o empleando expertos en computación en universidades o agencias gubernamentales para apoyarlos. Es un hecho que, en todos los países desarrollados una fracción importante del gasto en R&D (***R***esearch and ***D***evelopment – Investigación y Desarrollo) es pagada por el gobierno. Sin embargo, el gobierno debería restringirse a subsidiar *investigación básica*. La investigación básica es un bien público con fuertes externalidades positivas que no será provisto por el mercado. Típicamente, no es claro cuales serán los resultados de la investigación básica, cuanto tiempo llevará el descubrimiento, y cuales serán los usos. Los potenciales efectos derrame positivos son amplios y difíciles de internalizar por las firmas comerciales, por lo que estas tiene escasos incentivos financieros para este tipo de emprendimientos. Este es el motivo por el cual típicamente la investigación básica es llevada a cabo en universidades o laboratorios de investigación con financiación publica.

El desarrollo de la mayoría de los programas informáticos, sin embargo, corresponden a R&D aplicado y por lo tanto estos problemas son mucho menos severos. El R&D aplicado es dirigido al desarrollo de un producto especifico con ciertas características o a la solución de un problema tecnológico bien definido. Aquí, el resultado del proceso de R&D es mas predecible y es mucho mas fácil internalizar los efectos derrame positivos a través de protección con patentes o derechos de copyright y licenciando la nueva tecnología desarrollada a terceras partes. Por lo tanto, el R&D aplicado puede ser provisto por el mercado.

Mas aun, para el R&D aplicado es muy importante que el producto sea desarrollado orientándose hacia las necesidades de consumidores potenciales. Un firma maximizadora de beneficios tiene fuertes incentivos para hacer esto. Cuanto mejor se adapte su producto a las necesidades de sus clientes, mas consumidores estarán interesados en comprar su producto y mayor el precio que podría ser cobrado. Así, una compañía comercial se enfocará en lo que los consumidores quieren, y tratará de hacerlo de la manera más costo eficiente. Solo si desarrolla un mejor producto y a un costo mas bajo que sus competidores, será posible que sobreviva en el mercado. En contraste, un laboratorio de investigación financiado por el gobierno no enfrentará las restricciones del mercado y tiene muchos menos incentivos para focalizarse en las necesidades de los consumidores y perseguir la eficiencia de costos.

Otro problema con los subsidios públicos para R&D aplicado es que puede dar lugar al rent seeking. Un proyecto que quiere conseguir fondos públicos no tiene que superar a otros proyectos compitiendo en el mercado, si no que tiene que superarlos haciendo lobby para conseguir mayor apoyo político. A los

investigadores típicamente no les gusta hacer esto. Para que sus proyectos sobrevivan, a veces aceptan el apoyo de grandes compañías comerciales quienes pueden tener intereses algo distintos. Existen muchos ejemplos de proyectos científicos bien intencionados que fueron capturados por grandes compañías que posteriormente "se las arreglaron" para conseguir vastas sumas de subsidios públicos (por ejemplo, en las industrias de defensa, espacio y energía nuclear).

Si bien los argumentos hasta ahora enumerados aplican a todas las industrias, existen algunos argumentos adicionales importantes que surgen de las características específicas de la industria del software. A causa de las externalidades de red, un producto capturará la parte del león del mercado. Si los consumidores coordinan en el producto correcto, este resultado será eficiente. Sin embargo, cual producto sobrevivirá depende de las expectativas de consumidores que a su vez estarán afectadas por el comportamiento de grandes jugadores tales como el gobierno. Por lo tanto, aun subsidios modestos a un proyecto particular pueden tener un fuerte impacto en el resultado de mercado. En el extremo, el apoyo del gobierno a un proyecto puede causar que el mercado se mueva hacia otro equilibrio en el cual el producto apoyado por el gobierno desplace a otros productos del mercado. Pero el gobierno tiene una pobre historia de "picking winners" (escoger ganadores) a través de políticas industriales. El gobierno no tiene ni la habilidad ni los incentivos para decidir cual software es el más eficiente. Esto debiera ser dejado al mercado, y el gobierno debiera restringirse a proveer un campo de juego parejo.

Finalmente, si el gobierno decidiera subsidiar alguna investigación básica para el desarrollo de software, debiera asegurarse que esta investigación fuera ampliamente diseminada y que pudiera ser utilizada por todos.

En resumen, de acuerdo a lo expuesto en este apartado, este tipo de intervención gubernamental no sería recomendable para incentivar la producción de OSS.

2.2.1.2. Adopción de Software Open Source en el sector publico.

Varios gobiernos (por ejemplo, Brasil y Alemania) están actualmente considerando prohibir a las agencias gubernamentales utilizar software propietario si existe un software open source alternativo y forzar a escuelas y universidades a cambiar hacia el open source.

Los proponentes de esta política (tales como Ghosh y Schmidt, 2006 / Hahn, 2002) discuten que el FLOSS u OSS es cualitativamente mejor que el software propietario y que el costo total de propiedad (incluyendo el costo de mantenimiento y soporte técnico) es menor. Si este es el caso (que podría serlo al menos para algunas aplicaciones de software) entonces las agencias gubernamentales, firmas privadas, y consumidores; debieran estar felices de cambiar a OSS y no serían necesarias acciones coercitivas del gobierno para inducirlos a dicho cambio.

Algunos proponen el argumento adicional (German Bundestag, por ejemplo) que el OSS debiera ser adoptado para fortalecer una segunda fuente que ponga presión competitiva sobre el desarrollador de software propietario, induciéndolo a bajar precios.

Sin embargo, mientras que es claramente cierto que la existencia y rápido desarrollo del movimiento open source pone presión competitiva sobre los desarrolladores de software propietario y restringe su comportamiento en cuanto a precios, este solo hecho *per se* no implica que el gobierno debiera intervenir sobre el mercado favoreciendo el OSS, ya que lo anterior no necesariamente elevará el bienestar agregado de la sociedad: por ejemplo, la disminución en el excedente de los productores podría mas que compensar el aumento en el excedente de los consumidores.

2.2.1.3. Subsidios públicos para OSS con externalidades de red "fuertes".

Considérese primero el caso en que los efectos de red son fuertes en el sentido que solo un producto captura la parte del león en el mercado. En este caso, el gobierno podría hacer que el mercado vire. En particular, si fuerza a las escuelas y universidades a adoptar OSS, reducirá significativamente el costo de aprendizaje de los estudiantes para el uso de OSS. Si existen altos costos de switching para volver al software propietario posteriormente, el OSS puede capturar el mercado completo aun si no es cualitativamente superior al software propietario. Así, puede ser que el gobierno sea quien elige a los ganadores en este mercado. Segundo, si una parte significativa del mercado esta dirigida al open source, las ganancias presentas y futuras de las firmas de software propietario se verían reducidas. Por lo tanto, tendrían menores incentivos a innovar. Tercero, esta política puede desalentar la entrada de nuevos desarrolladores de software propietario si estos sienten que no pueden competir con el OSS que es favorecido por el gobierno. Finalmente, si el OSS favorecido por el gobierno tiene licencia GPL u otra licencia viral, la cual hace legalmente difícil para los desarrolladores de software propietario hacer su software compatible con el OSS, entonces el gobierno estaría impulsando una estructura de desarrollo en la cual habría dos redes incompatibles en el mercado. Esto es ineficiente, dado que significativos efectos de red positivos se perderían, y reduciría la competencia en el mercado.

2.2.1.4. Subsidios públicos para OSS con externalidades de red "débiles".

Un caso intermedio sería aquel en el cual los efectos de red son débiles y/o los programas informáticos (software) son altamente compatibles entre si de modo que dos o más productos pueden sobrevivir en el mercado a largo plazo. Aun en este caso, un movimiento a OSS del gobierno no necesariamente incrementara la competencia y reducirá los precios. En realidad, en circunstancias normales, sucedería lo exactamente opuesto.

Supóngase que existen dos software distintos en el mercado. Uno es propietario vendido por un desarrollador maximizador de beneficios, mientras que el otro es OSS entregado gratuitamente. Se podrían distinguir tres tipos de consumidores en este mercado: algunos consumidores siempre comprarán el software propietario (por ejemplo, porque son usuarios poco sofisticados que se reconocen sin los conocimientos necesarios para utilizar OSS), algunos consumidores siempre irán tras el OSS (por ejemplo, porque quieren adaptar el software a sus necesidades especificas) y otros consumidores considerarán ambos tipos de software y basarán su decisión en las respectivas cualidades, el precio del software propietario y sus preferencias ideosincraticas.

Supóngase ahora que el gobierno fuerza a algunas agencias gubernamentales (quienes pertenecen al tercer grupo) a adoptar OSS. Esto reduce el tamaño del mercado para aquellos consumidores para quienes hay competencia. Así, el desarrollador de software propietario tendrá menos incentivos para reducir sus precios y competir por estos consumidores, si no que podría ser que aumente sus precios para alcanzar mayores ganancias entre los consumidores del primer grupo. Por lo tanto, el precio del software propietario aumentaría. Como consecuencia, consumidores, agencias del gobierno, y desarrolladores de software propietario, estarán peor: los consumidores tiene que pagar mayores precios por el software propietario (mientras el precio del OSS permanece inalterado), las agencias gubernamentales ya no tienen opción si no que son forzadas a adoptar OSS, y el desarrollador de software propietario pierde parte de sus ganancias. Por otro lado, el desarrollador de OSS no se beneficia de esta política en términos financieros, porque continúa ofreciendo su producto a precio cero.

En este contexto, los incentivos a innovar para el desarrollador de software propietario se reducen, porque el tamaño de mercado de los consumidores indecisos, para los cuales hay competencia vía calidad, se reduce. Así, una mejora cualitativa inducirá a menos consumidores a cambiar al software propietario, siendo menos rentable para el desarrollador de software propietario competir en calidad. Al mismo tiempo, los incentivos de los desarrolladores de OSS no se ven afectados en forma directa. El tamaño de su mercado ha aumentado, pero las mejoras cualitativas no redundarán en mayores beneficios financieros.

En conclusión, si las externalidades de red son débiles y existe competencia para un software particular, la intervención del gobierno imponiendo la utilización de OSS probablemente reduzca la competencia, incremente precios, reduzca los incentivos a innovar, y todo el mundo este peor.

2.2.1.5. Subsidios para instituciones que coordinan el desarrollo de FLOSS.

El gobierno podría subsidiar instituciones del movimiento open source que tratan de coordinar el desarrollo de software y estándares. Un ejemplo es Berlios, un mediador entre desarrolladores y consumidores, co – fundado entre el gobierno federal alemán y compañías privadas como Hewlett

Packard y Linux Information Systems. Esta política puede tener algunos meritos, en particular si el rol de tales instituciones es neutral respecto a cualquier proyecto open source particular y restringido a alentar estándares abiertos y compatibilidad entre software open source y propietario.

Sin embargo, el problema básico de cualquier intervención gubernamental permanece. No es asunto del gobierno decidir que software será el estándar en el futuro. Más aún, cualquier apoyo financiero a proyectos OSS particulares invitará a actividades de rent seeking, y la decisión de que proyecto promover estará sesgada por presiones políticas.

Así, aunque podrían existir justificativos para este tipo de intervención oficial, podrían darse situaciones que ameriten la no recomendación de la misma.

2.2.2. Modelización con discriminación entre usuarios informados y no informados.

El trabajo de Comino y Manenti (Open Source vs. Closed Source Software: Public Policies in the Software Market. Stefano Comino & Fabio M. Manenti. June 2003), plantea una visión con mayores posibilidades para la intervención publica en apoyo del desarrollo de OSS. Diferenciándose de lo planteado por Schmidt y Schnitzer, incorporan a su marco teórico elementos no incluidos por estos autores.

Comino y Manenti reconocen la presencia de una fracción importante de usuarios no sofisticados o "no informados", especialmente en el segmento de difusión masiva, siendo esto una característica intrínseca de la industria del software. Mientras que el software propietario es producido y vendido por firmas comerciales con fuertes incentivos para publicitar sus productos e informar a usuarios potenciales acerca de sus "paquetes", el OSS es generalmente distribuido por desarrolladores individuales o grupos, con motivaciones distintas a la maximización de beneficios.

Partiendo de estas simples observaciones, se analizaran los efectos que tienen sobre el bienestar las distintas formas de intervención oficial que han sido planteadas por varios gobiernos en el mundo.

2.2.2.1. Adopción mandatoria.

Solo si la masa de consumidores no informados es suficientemente grande, será óptimo imponer la adopción de software open source (OSS).[1]

[1] *Demostración disponible en Comino & Manenti 2003.*

Dado que la masa de potenciales usuarios del software propietario se ve reducida, las ganancias de las firmas comerciales declinan. El efecto sobre los excedentes del consumidor depende de la tipología de los usuarios. El excedente de aquellos consumidores que no han sido obligados a adoptar OSS no se ve alterado. Los consumidores informados que se ven ahora obligados a adoptar OSS estarán peor: aquellos que cambian del software propietario al OSS como consecuencia de la política obtendrán una utilidad estrictamente menor, mientras que aquellos que ya se encontraban adoptando OSS no serían afectados por la política. El impacto sobre los consumidores no informados obligados por el gobierno dependerá de su situación: mientras que la política perjudicara a aquellos con fuertes preferencias por el software propietario, los usuarios no informados que sin la política no comprarían ningún software estarán mejor. En promedio, el último efecto (positivo) domina al primero.

De esta discusión, es claro que solo aquellos usuarios no informados que son obligados por el gobierno a adoptar OSS se beneficiarían de esta política. Por lo tanto, la adopción mandatoria es incrementadora del bienestar solo cuando la masa de usuarios no informados es suficientemente grande. Este resultado contrasta con el obtenido por Schmidt y Schnitzer (2003) en el cual la adopción mandatoria siempre lastima a la sociedad; esto se debe al fuerte supuesto implícito que mientras la firma no es capaz de discriminar entre usuarios informados y no informados, el gobierno discrimina a los consumidores y selecciona solo a los informados al imponer la adopción de OSS.

Por lo tanto, bajo el esquema teórico propuesto por Comino y Manenti (Open Source vs. Closed Source Software: Public Policies in the Software Market. Stefano Comino & Fabio M. Manenti. June 2003), esta forma de intervención sería una opción viable siempre que la masa de consumidores no informados sea lo suficientemente grande.

2.2.2.2. Campañas informativas.

A través de una *campaña informativa*, algunos consumidores no informados reciben información acerca de la existencia y características del OSS. Se asumirá para este análisis que el gobierno implementa tales políticas sin costo.

De acuerdo a Comino y Manenti (Open Source vs. Closed Source Software: Public Policies in the Software Market. Stefano Comino & Fabio M. Manenti. June 2003), *apoyar el software open source (OSS) a través de campañas informativas sin costo siempre incrementa el bienestar (ante la presencia de externalidades de red, esta conclusión se mantiene solo si la campaña es drástica).*[2]

[2] *Demostración disponible en Comino & Manenti 2003.*

Como en el caso de la adopción mandatoria, el efecto de la política sobre el excedente de los productores es directo: dado que la política reduce el mercado cautivo de usuarios no informados, sin duda perjudica a la firma.

Para entender el efecto de las campañas informativas sobre el excedente de los consumidores es necesario distinguir entre dos casos dependiendo de si el precio cobrado por el software propietario disminuye o se incrementa con la proporción de usuarios que pasa de ser no informados a informados. Suponiendo que este precio cae, entonces todos los consumidores estarán mejor. Más precisamente, mientras que los consumidores informados que adoptan el software propietario se benefician de precios más bajos, la política no tiene impacto sobre aquellos que adoptan OSS. Los consumidores no informados que adoptan software propietario se benefician de la reducción de precios mientras que aquellos que no adoptan ningún software no se ven afectados. Finalmente, todos los usuarios no informados que se convierten en informados a través de la campaña informativa, estarán mejor. Si adoptan el software propietario se les cobrará un precio menor, mientras que si adoptan el OSS se benefician de una opción que no estaba disponible antes.

De la discusión anterior es claro que cuando la política induce a la firma a cobrar un precio menor, la campaña informativa hace que el mercado funcione mejor en todos los aspectos; todas las fallas de mercado son suavizadas por esta política: incrementa la proporción de consumidores que adoptan el software propietario tanto en el segmento de usuarios informados como no informados, e induce a algunos usuarios previamente no informados a tomar una decisión mas apropiada. Por lo tanto, no resulta sorprendente que el bienestar total se incremente con el tamaño de la campaña informativa (la proporción de usuarios no informados que se convierten en informados a través de la campaña).

Sin embargo, la campaña informativa siempre incrementa el bienestar, aun cuando induce un aumento en el precio del software propietario. Aquí, el hecho que una fracción de los consumidores sea ahora capaz de realizar una decisión de adopción mas informada, hace que la política en estudio sea siempre incrementadora de bienestar, lo cual convierte a esta política en una de las mas atractivas desde el punto de vista teórico (aunque esto no necesariamente implica que sea la de mas factible implementación).

2.2.2.3. Subsidios.

Se supondrá en este análisis que el gobierno paga una transferencia monetaria a cada individuo que adopte el OSS.

Existe consenso, tanto entre autores en contra como a favor del apoyo oficial al movimiento OSS, respecto a que subsidiar en forma directa dicho movimiento reduce el bienestar.[3]

Esta política claramente induce al desarrollador de software propietario a cobrar un precio diferente por sus productos. No obstante, para tener una intuición mas clara de la proposición anterior, considérese primero que ocurriría si el precio no cambiara. La utilidad de aquellos consumidores que adopten OSS aumentaría en la suma del subsidio. Sin embargo, este efecto positivo es exactamente esterilizado por el incremento en el gasto del gobierno y por lo tanto, para estos consumidores no habría efecto neto sobre el bienestar. Similarmente, también para aquellos consumidores que adoptan el software propietario no habría efecto sobre el bienestar social: su utilidad no cambia y el gobierno no les paga subsidio alguno. Por el contrario, para aquellos consumidores que cambian del software propietario al OSS como consecuencia de la política, el efecto neto sobre el bienestar es estrictamente negativo. El incremento en la utilidad de estos consumidores no compensa el incremento en los gastos del gobierno. Sin los subsidios, el software propietario sería su elección, y al moverse al OSS obtienen una utilidad mayor pero el incremento es menor que el monto del subsidio.

Finalmente, la política lastima al desarrollador de software propietario; es posible demostrar que la firma tiene que recortar sus precios para proteger su mercado. Mientras que los consumidores se benefician de esta baja de precios, el efecto global del subsidio es negativo.

El análisis hasta ahora detallado, solo consideró la presencia de externalidades de red al estudiar el subsidio directo bajo el modelo teórico de Schmidt y Schnitzer (Schmidt & Schnitzer 2002). Es decir, el efecto de esta externalidad solo se incorporó bajo un esquema que buscaría mostrar la ineficacia de la intervención pública a favor del OSS.

Como se mencionara previamente, el modelo de Comino y Manenti (Comino & Manenti 2003), brinda un marco teórico mas general al levantar el fuerte supuesto implícito que mientras la firma no es capaz de discriminar entre usuarios informados y no informados, el gobierno discrimina a los consumidores y selecciona solo a los informados al imponer la adopción de OSS. Al mismo tiempo, Comino y Manenti reconocen la diferencia de incentivos para los desarrolladores de software OSS respecto a los del software propietario.

Al incorporar la posibilidad de existencia de externalidades de red en este marco más general, se ven afectados los resultados obtenidos, con su consecuente impacto sobre las recomendaciones para políticas públicas surgidas de este análisis.

[3] *Demostración disponible en Comino & Manenti 2003.*

La presencia de externalidades de red en el mercado de software implicaría que el beneficio individual de la adopción de un cierto software es afectado positivamente por el número de otros individuos que han adoptado el mismo software o uno compatible. Este efecto se funda en la simple observación que cuanto mas expandido sea un paquete de software, mas fácil será el intercambio de archivos e información con otros usuarios y, por lo tanto mayor será la utilidad de adoptar ese software en particular.

La presencia de pequeñas externalidades de red, hace a la política mas efectiva (o equivalentemente, menos dañina) si existen pocos usuarios no informados; con mayores proporciones de usuarios no informados, esto ocurre solo si el gobierno impone la adopción de OSS (mandatoriamente o a través de campañas informativas) a un numero de usuarios lo suficientemente grande.

Ante la presencia de externalidades de red, el bienestar aumenta en tanto el mercado se incline hacia la estandarización en un software: en este caso la externalidad de red, y por lo tanto el bienestar, están en su máximo nivel.

Cuando la vasta mayoría corresponde a usuarios informados, dado que el OSS es gratis, este gana una base instalada mayor que la del software propietario. Al imponer la adopción de OSS a usuarios adicionales (mandatoriamente o a través de campañas informativas), el gobierno inclina el mercado aun más hacia el software open source; esto refuerza el efecto positivo sobre el bienestar de una base instalada mayor y mejora la performance de la política. Por lo tanto, como se refirió anteriormente, cuando la proporción de usuarios no informados es pequeña, la presencia de externalidades de red hace que la adopción mandatoria o la campaña informativa sea mas efectiva (o menos dañina en el caso que la política tenga un efecto total negativo sobre el bienestar) en mejorar el bienestar social. Por el contrario, cuando el número de usuarios no informados es lo suficientemente grande, entonces el OSS ya no tendrá la mayor base instalada. Como consecuencia, la presencia de externalidades de red mejorará la efectividad de la política solo si el gobierno es capaz de influir la decisión de adopción de un gran número de consumidores haciendo así que el OSS sea el software con la mayor base instalada.

2.2.3. Modelos con mayor nivel de abstracción.

Una aproximación alternativa al rol jugado por las externalidades de red y los posibles roles a asumir por el Estado, puede encontrarse en modelizaciones más generales, tales como la propuesta por Church y Gandal (Network Effects, Software Provision and Standarization. Jeffrey Church & Neil Gandal. March 1992).

A continuación, se brinda un acercamiento a los aspectos esenciales del modelo citado en el párrafo precedente.

2.2.3.1. Preferencias de los consumidores.

Se considera una función de utilidad CES ($1 < \beta < 2$):

$U(x_1, x_2, ..., x_N) = \left(\sum_i x_i^{1/\beta} \right)^{\beta} + \phi$, asumiéndose así, que la variedad es importante.

La distribución del gusto de los consumidores en cuanto a la tecnología de hardware es uniforme a lo largo de una línea de longitud unitaria, densidad normalizada a uno.

El consumidor representativo, adquiere un sistema que maximiza su utilidad (*k* mide el grado de diferenciación de producto entre tecnologías de hardware):

$U(x_1, x_2, ..., x_N) = \left(\sum_i x_i^{1/\beta} \right)^{\beta} + \phi - kt$, sujeto a la siguiente restricción presupuestaria:

$\sum_i \rho_i x_i = y - p_h$, donde ρ_i es el precio de la variedad *i* del software en la red *h = A, B*; *y* es el gasto total alocado en hardware y software, y p_h es el precio de una unidad de hardware para el sistema *h*.

Tal como se deriva en Church y Gandal (Church & Gandal, 1989), el sistema de ecuaciones de demanda, para todo *i*:

$x_i[\rho_i, p_h, q_h] = (y - p_h)(q_h)^{1/(\beta-1)}(\rho_i)^{\beta/(\beta-1)}$, donde $q_h(\rho_1, \rho_2, ..., \rho_N) = \left[\sum_j (\rho_j)^{1(1-\beta)} \right]^{(1-\beta)}$

Cuando el precio de cada marca de software es el mismo ρ se obtiene $q_h = \rho N^{(1-\beta)}$, y por lo tanto:

$V(q_h, p_{h,}, N, t) = \frac{N^{\theta}(y - p_h)}{\rho} + \phi - kt$, donde $\theta = \beta - 1$.

La función de utilidad indirecta es creciente respecto al número de productos de software y cóncava para $0 < \theta < 1$ (θ mide la preferencia del consumidor por la variedad de software).

La utilidad de reserva de la opción externa es normalizada a cero.

Cada consumidor compra uno de los sistemas de hardware que compite y algunos software desarrollados para ese sistema. Dado que $\theta > k$, un consumidor situado en un extremo de la línea, adquiere un sistema localizado en el otro extremo de la línea, si no hay alternativa.

2.2.3.2. Tecnología.

Se asume que las tecnologías de hardware son incompatibles y no propietarias, ofrecidas a un costo marginal *c*.

Cada firma de software puede producir solo un producto y debe elegir a que red proveerá software.

La tecnología para producción de software esta caracterizada por rendimientos crecientes a escala (costo marginal de producción del software constante – *s* y costo de desarrollo del software también constante – *F*).

Se asume a las firmas de software como competidores monopolísticos, lo cual implica el precio de equilibrio del software $\rho = \beta s$ (derivado en Church y Gandal (Church & Gandal, 1991)), de modo que el precio del software no depende del número de firmas.

2.2.3.3. Timing del juego.

Las firmas de software entran a la industria (libre entrada), teniendo las mismas expectativas racionales sobre el número software producido para cada tecnología.

Para un conjunto dado de parámetros (*y, c, F,* β), el número de firmas de libre entrada $N_f = N_A + N_B$ es único.

Las firmas de software eligen simultáneamente a que plataforma proveerán software, para luego incurrir en los costos de desarrollo del software.

Los consumidores, compran uno de los dos sistemas de hardware incompatibles y algunos de los productos de software producidos para ese sistema.

2.2.3.4. Efectos de red.

Sea el consumidor indiferente:

$$V(\rho, p_A = c, N_A, t) = V(\rho, p_B = c, N_B, 1-t)$$

$$(y-c)\frac{N_A^\theta}{\beta s} + \phi - kt = (y-c)\frac{N_B^\theta}{\beta s} + \phi - k(1-t)$$

Se asume que si no existen productos de software próximos a aparecer, un consumidor no comprara la tecnología de hardware (si $N_A = 0$ entonces $t = 0$, si $N_B = 0$ entonces $t = 1$).

Resolviendo por *t*, se obtiene:

$$t(N_A) = \frac{N_A^\theta (y-c) - (N_f - N_A)^\theta (y-c) + k\beta s}{2k\beta s}$$

Lema: El share de una red, es creciente respecto al número de productos de software se esa red, provisto que $0 < t < 1$.

Para N_A se obtiene:

$\frac{dt}{dN_A} = \frac{\theta N_A^{(\theta-1)}(y-c) - \theta(N_f - N_A)^{(\theta-1)}(y-c)}{2k\beta s} > 0$, y análogamente para B.

La expresión $\frac{dt}{dN_A}$ representa el efecto de red.

2.2.3.5. Reglas de estandarización.

Proposición 1:

- *Caso 1: Cuando* $N_f^\theta < \frac{k\beta s}{(y-c)}$ ambas tecnologías tendrán ventas $(0 < t < 1)$ para todas las posibles configuraciones de software, excepto $(N_A = N_f)$ y $(N_B = N_f)$. Para $(N_A = N_f)$ y $(N_B = N_f)$ una tecnología con el conjunto completo de firmas de software, representa el estándar de mercado.
- *Caso 2:* Cuando $N_f^\theta > \frac{k\beta s}{(y-c)}$ existe un número critico de firmas de software para cada tecnología, $\eta < N_f$, tal que cuando el número de productos de software provistos para una red, *h*

es mayor o igual a η, la tecnología *h* conforma el estándar de mercado; es decir, si $N_A \geq \eta$ o $N_B \geq \eta$ entonces $t = 1$ o $t = 0$.

2.2.3.6. Firmas de software.

La ganancia de las firmas de software que desarrollan un producto para la red A, es:

$\Pi_A = t(\rho - s)x - F$, donde $x = \frac{(y-c)}{\rho N_A}$ y $\rho = \beta s$, así:

$$\Pi_A[t, p_A = c, N_A] = \frac{(\beta - 1)t(y-c)}{[\beta N_A]} - F$$

$$\Pi_B[t, p_B = c, N_B] = \frac{(\beta - 1)t(y-c)}{[\beta N_B]} - F$$

En la segunda etapa, las firmas de software simultáneamente eligen a que red proveerán de software.

A fin que una división de las firmas de software sobre las redes sea un equilibrio de Nash, no puede ser posible para la firma incrementar sus ganancias al saltar entre las redes. Cuando una división de equilibrio de las firmas de software es obtenida, se incurre en los costos fijos.

N_A afecta las ganancias a través de un efecto directo y un efecto sobre el market share:

$$\frac{d\Pi_A}{dN_A} = \frac{(\beta - 1)(y-c)}{\beta N_A}\left(\frac{dt}{dN_A} - \frac{t}{N_A}\right)$$

2.2.3.7. Solución de equilibrio.

Proposición 2:

- *Caso 1:* Si $N_f^\theta < \frac{k\beta s}{(y-c)}$, existirá un equilibrio simétrico único $\left(N_A^* = \frac{N_f}{2}, N_A^* = \frac{N_f}{2}\right)$. Ambas redes siempre existirán en esta región del espacio de parámetros.
- *Caso 2:* Si $N_f^\theta > \frac{k\beta s}{[2\theta(y-c)]}$, dos equilibrios, $\left(N_A^* = N_f, N_B^* = 0\right)$ y $\left(N_A^* = 0, N_B^* = N_f\right)$ existirán. Una estandarización *de facto* ocurre en esta región del espacio de parámetros.

- *Caso 3:* Si $\frac{2^{\theta} k\beta s}{[2\theta(y-c)]} \geq N_f^{\theta} \geq \frac{k\beta s}{(y-c)}$, existirán tres equilibrios candidatos; $\left(N_A^* = \frac{N_f}{2}, N_B^* = \frac{N_f}{2}\right)$, $\left(N_A^* = N_f, N_B^* = 0\right)$ y $\left(N_A^* = 0, N_B^* = N_f\right)$. Se define $c_1 > c_2$ tal que $N_f^{\theta} = \frac{k\beta s}{(y-c_1)}$ y $N_f^{\theta} = \frac{2^{\theta} k\beta s}{[2\theta(y-c_2)]}$. Existirán valores críticos c_i y c_s tales que, para $c_i > c \geq c_2$ el equilibrio de estandarización existe, para $c_s \geq c \geq c_i$ los tres equilibrios existen, y para $c_1 \geq c > c_s$ el equilibrio $\left(N_A^* = \frac{N_f}{2}, N_B^* = \frac{N_f}{2}\right)$ existe.

El número de firmas de libre entrada es único (condición de ganancia marginal cero): $N_f = \frac{(y-c)(\beta-1)}{\beta F}$.

2.2.3.8. Bienestar.

El bienestar total será mayor bajo estandarización siempre que el bienestar del consumidor sea mayor bajo estandarización.

La ganancia para los consumidores derivada de la estandarización es un incremento en el número de variedades de software disponible, mientras que el costo de estandarización es una reducción en la variedad de hardware disponible.

Para múltiples sistemas de equilibrio, la utilidad indirecta de un consumidor ubicado en *t* es:

$$V\left(\rho = \beta s, p = c, N = \frac{N_f}{2}, t\right) = \left(\frac{N_f}{2}\right)^{\theta} \frac{(y-c)}{\beta s} + \phi - kt$$

$$2\int_0^{1/2}\left[\left(\frac{N_f}{2}\right)^{\theta} \frac{(y-c)}{\beta s} + \phi - kt\right]dt = \left(\frac{N_f}{2}\right)^{\theta} \frac{(y-c)}{\beta s} + \phi - \frac{k}{4}$$

Si un planificador impone la adopción de una tecnología, la función de utilidad indirecta para un consumidor localizado en *t,* es:

$$V\left(\rho = \beta s, p = c, N = N_f, t\right) = N_f{}^{\theta} \frac{(y-c)}{\beta s} + \phi - kt$$

$$\int_0^1 \left[\left(\frac{N_f}{2} \right)^{\theta} \frac{(y-c)}{\beta s} + \phi - kt \right] dt = N_f^{\theta} \frac{(y-c)}{\beta s} + \phi - \frac{k}{2}$$

Proposición 3:

Un regulador que maximiza el bienestar total impondrá la estandarización siempre que $N_f^{\theta} > \frac{k\beta s}{\left[4\left(1 - \left(\frac{1}{2} \right)^{\theta} \right)(y-c) \right]}$

Proposición 4:

Si los consumidores valoran relativamente alto la variedad, de modo que $\theta > \frac{\log\left(\frac{3}{4}\right)}{\log\left(\frac{1}{2}\right)}$, entonces existirán valores para los parámetros tal que ambas redes operarán en el mercado, pero el bienestar total sería mayor bajo adopción mandatoria de una única tecnología. Así, se tiene que:

- *El resultado de mercado entraña un nivel sub-optimo de estandarización para varios valores de los parámetros.*
- *La dimensión de la ineficiencia crece al incrementarse el valor asignado a la variedad.*
- *El bienestar total se incrementaría si un estándar fuera impuesto.*

2.3. El rol del estado.

De existir, los usuarios de software se verán afectados de manera significativa por las externalidades de red directas. El ejemplo clásico es el intercambio de archivos. Cualquiera que tenga el mismo programa para correr un tipo de archivo determinado puede leer y modificar archivos recibidos de otra gente quienes, a su vez, pueden hacer lo mismo. De aquí la importancia de compartir un tipo particular de software: el sistema operativo sobre el cual corre el sistema completo de archivo de la computadora.

Las externalidades de red influencian profundamente el proceso de difusión que, en tal caso, corresponde al problema de surgimiento de estándares (Arthur, 1989, 1994, 1996). En estos casos una pequeña ventaja que por casualidad ocurriera al comienzo del proceso de difusión es todo lo que se necesita para favorecer un estándar que de ese modo conquista la totalidad del mercado, que permanecerá posteriormente *locked – in* (*path dependence*).

La tesis del efecto *lock – in* ha sido materia de acalorados debates. Interesantemente, la naturaleza de la tecnología Open Source permite la formulación de un número de puntos relevantes a este debate.

El análisis de estándares en la teoría económica enfatizan cuan fácil resulta caer en la trampa bien ilustrada por el análisis de Paul David de la batalla Qwerty – Dvorak: "Si solo existen teclados Qwerty, quienes tipean estudiarán solo Qwerty, pero si quienes tipean estudian solo Qwerty, solo habrá teclados Qwerty". Para explicar esta dinámica, David (David, 1985) refiere al trabajo de Arthur (Arthur, 1989, 1994) acerca de cómo los rendimientos crecientes operan en el proceso de difusión de dos tecnologías que compiten, donde al ser más utilizada una determinada tecnología, mayor será la probabilidad que así continúe en el futuro. Los rendimientos crecientes son una consecuencia inmediata de las externalidades de red, al punto que Witt (Witt, 1997) discute que ambas pueden ser tratadas como sinónimos.

Dada la presencia en el proceso de difusión del software de las previamente descritas externalidades de red directas, el mecanismo de *lock – in* parece un resultado inevitable: si una pieza de software logra ganar un *market share* significativo, un circulo virtuoso se pone el marcha tal que los consumidores tendrán aún más incentivos a usarlo, habrá un incremento en la oferta de productos complementarios (aplicaciones, mantenimiento, etc.) y esa pieza de software particular comenzará a dominar el mercado consolidándose en estándar.

El concepto puede ser formalizado utilizando la teoría de las urnas de Polya (Arthur et al., 1993). El esquema es el muestreo repetitivo de una urna conteniendo pelotas rojas y azules, en la cual cada vez que se extrae una pelota de un color, otra del mismo color es agregada. En cada periodo *t*, la probabilidad de agregar una pelota de color rojo es una función creciente de la proporción de pelotas rojas en la urna, aplicándose un razonamiento análogo para las pelotas azules. El modelo muestra que cuando el número de pelotas se extiende a infinito, la proporción de pelotas de un mismo color converge en probabilidad a uno. Habiendo explicado el efecto *lock – in*, resta explicar, sin embargo, que lo desencadena. Arthur (Arthur, 1989) refiere a la presencia de "pequeños eventos" que "de casualidad, pueden dar a una de las dos tecnologías la ventaja necesaria". Si tal es la situación, no existe defensa contra la posibilidad que tecnologías ineficientes establezcan su dominancia a expensas de otra, que, de adoptarse, sería superior.

La existencia de maza crítica en el esparcimiento de una nueva tecnología, es así crucial para que se dé un cambio de estándar. De acuerdo a Schelling (Schelling, 1978), el término se tomó de la física donde se refería a la cantidad de combustible radiactivo requerido para lograr fisión nuclear. Transportando este concepto a las esferas económica y social, se habla de "efecto maza crítica" si, cuando ciertas variables que caracterizan un proceso se elevan por encima de determinado umbral, el fenómeno explota, de modo que el sistema se mueve del equilibrio estable en el cual se posicionaba al inicio y asume otro equilibrio

estable. En los modelos de difusión tecnológica, esta variable es representada por el número de personas que adoptan la nueva tecnología. (Bonaccorsi & Rossi, 2002).

Así, aún cuando el estándar de interfases cerradas y propietarias sea cualitativamente inferior, continuará manteniendo su predominancia a menos que el Estado actúe coordinando la formación de base instalada de un estándar alternativo, tal como el planteado por el OSS.

Es más, los resultados expuestos en secciones precedentes respecto a la sub – optimalidad de favorecer (aún con la no – intervención) la coexistencia de más de un estándar ante la presencia externalidades de red, implican que estas acciones tendientes a fomentar la formación de base instalada de un nuevo estándar cualitativamente superior, debieran ser lo suficientemente drásticas, de modo que se busque lograr un único estándar.

No obstante, debe considerarse que existirían importantes costos asociados a las medidas adoptadas tendientes a apoyar la creación de base instalada para el movimiento FLOSS, fundamentalmente referidos a la migración de estándares.

2.4. Implicancias.

De la discusión detallada en esta sección, surge que entender la viabilidad del crecimiento del OSS en países en desarrollo alentado mediante políticas gubernamentales, implicaría la necesidad de analizar si existirían, y como existirían, las referidas externalidades de red en estos países.

Un aspecto que resulta importante señalar, es que a nivel *macroeconómico,* podrían existir ventajas resultantes de la promoción y difusión del FLOSS, que no serían captados a partir de un análisis de tipo *microfundado*, tal como el expuesto.

3. ASPECTOS METÓDOLOGICOS DE LA INVESTIGACIÓN.

De acuerdo a lo analizado en la sección precedente, si existen externalidades de red significativas, la adopción mandatoria y las campañas informativas incrementan el bienestar solo si son "drásticas": cuando el gobierno es capaz de influir sobre una gran masa de consumidores entonces el mercado se inclina hacia el software open source y las externalidades de red alcanzan su máximo incrementando el bienestar.

A fin de estudiar la presencia de externalidades de red en la industria del software, se tomara como segmento de análisis el mercado de webservers.

La investigación se concentrara en la evolución de los market share correspondientes a distintos proveedores en el mercado mundial y el mercado argentino.

Dado que las teorías analizadas y las implicancias de ellas derivadas, tienen su origen en observaciones referidas a mercados desarrollados, es importante entender si existen diferencias sustanciales entre los resultados obtenidos para las series correspondientes a la Argentina y el mercado mundial (en el cual la ponderación de los mercados desarrollados, es dominante). A este fin, se compararán ambas estimaciones .

Las series a consideradas, pertenecen a los principales competidores en este mercado: Apache (APACHE) y Microsoft (MSFT); dentro de un marco metodológico econométrico de modelos VAR.

3.1. Análisis VAR.

Cuando no se tiene certeza acerca de la exogeneidad de una variable, una alternativa natural es tratar cada variable del modelo de manera simétrica. A fines de facilitar la exposición, se renombrarán las variables relevantes, siendo y_t: APACHE y z_t: MSFT.

Al permitir que el sendero temporal de la secuencia de y_t sea afectado por realizaciones pasadas y presentes de la secuencia de z_t, y viceversa; se tiene:

$$y_t = b_{10} - b_{12} z_t + \gamma_{11} y_{t-1} + \gamma_{12} z_{t-1} + \varepsilon_{yt} \quad (1)$$
$$z_t = b_{20} - b_{21} y_t + \gamma_{21} y_{t-1} + \gamma_{22} z_{t-1} + \varepsilon_{zt} \quad (2)$$

Donde se asume: (1) que ambas variables y_t y z_t son estacionarias; (2) que ε_{yt} y ε_{zt} son ruido blanco con desvío estándar σ_y y σ_z respectivamente y (3) ε_{yt} y ε_{zt} están incorrelacionados.

Las ecuaciones (1) y (2) constituyen un vector de autoregresión de primer orden (VAR 1) dado que el rezago más grande es 1. Este simple VAR de primer orden y dos variables, es útil para ilustrar sistemas multivariados de orden superior.

La estructura del sistema incorpora retroalimentación, ya que y_t y z_t se afectan mutuamente. Por ejemplo, $-b_{12}$ es el efecto contemporáneo de un cambio unitario de z_t sobre y_t y γ_{21} el efecto de un cambio unitario en y_{t-1} sobre z_t. Nótese que los términos ε_{yt} y ε_{zt} son shocks puros en y_t y z_t respectivamente. Si b_{21} no es cero, ε_{yt} tiene un efecto contemporáneo indirecto sobre z_t, y si b_{12} no es cero, ε_{zt} tiene un efecto contemporáneo indirecto sobre y_t.

Utilizando álgebra matricial, se puede re-escribir el sistema en forma compacta:

$$\begin{bmatrix} 1 & b_{12} \\ b_{21} & 1 \end{bmatrix} \begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} b_{10} \\ b_{20} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix}$$

o $Bx_t = \Gamma_0 + \Gamma_1 x_{t-1} + \varepsilon_t$

donde

$$B = \begin{bmatrix} 1 & b_{12} \\ b_{21} & 1 \end{bmatrix} \qquad x_t = \begin{bmatrix} y_t \\ z_t \end{bmatrix} \qquad \Gamma_0 = \begin{bmatrix} b_{10} \\ b_{20} \end{bmatrix} \qquad \Gamma_1 = \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}$$

$$\varepsilon_t = \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix}$$

Premultiplicando por B^{-1} permite obtener el modelo VAR en forma estándar:

$$x_t = A_0 + A_1 x_{t-1} + e_t (3)$$

donde:

$$A_0 = B^{-1}\Gamma_0$$

$$A_1 = B^{-1}\Gamma_1$$

$$e_1 = B^{-1}\varepsilon_t$$

Por conveniencia notacional, se define a_{i0} como el elemento i del vector A_0, a_{ij} como el elemento de la fila i y la columna j de la matriz A_1, y e_{it} como el elemento i del vector e_t. Usando esta nueva notación, se puede re-escribir (3) en la forma equivalente:

$$y_t = a_{10} + a_{11}y_{t-1} + a_{12}z_{t-1} + e_{1t} \quad (4)$$
$$z_t = a_{20} + a_{21}y_{t-1} + a_{22}z_{t-1} + e_{2t} \quad (5)$$

Para distinguir entre los sistemas representados por (1) y (2) versus (4) y (5), el primero es llamado ***VAR estructural*** o sistema primitivo y el segundo es llamado ***VAR en forma estándar***.

Es importante señalar que los términos de error (e_{1t} y e_{2t}) están compuestos por los dos shocks ε_{yt} y ε_{zt}. Dado que $e_t = B^{-1}\varepsilon_t$, se pueden calcular e_{1t} y e_{2t} de la siguiente manera:

$$e_{1t} = (\varepsilon_{yt} - b_{12}\varepsilon_{zt})/(1 - b_{12}b_{21}) \quad (6)$$
$$e_{2t} = (\varepsilon_{zt} - b_{21}\varepsilon_{yt})/(1 - b_{12}b_{21}) \quad (7)$$

3.2. Estimación.

A partir de la critica de Sims (1980) a lo que consideró "restricciones increíbles para la identificación", inherentes a los modelos estructurales, se genera una estrategia de estimación alternativa. Considérese la siguiente generalización multivariada de (3):

$$x_t = A_0 + A_1x_{t-1} + ... + A_px_{t-p} + e_t \quad (8)$$

donde:

x_t = Vector de $(n \times 1)$ conteniendo cada una de las n variables incluidas en el VAR.

A_0 = Vector de $(n \times 1)$ términos de intercepto.

A_i = Matrices de coeficientes de $(n \times n)$.

e_t = Vector de $(n \times 1)$ conteniendo términos de error.

La metodología de Sims entraña poco más que la determinación de las variables apropiadas a incluir en el VAR y la cantidad de rezagos. Las variables a incluir en el VAR son seleccionadas de acuerdo al

modelo económico relevante. Los test de longitud de rezago son utilizados para determinar la cantidad de rezagos a incluir. No se sugieren otros criterios para "disminuir" la cantidad de parámetros a estimar. La matriz A_0 contiene n términos de intercepto y cada matriz A_i contiene n^2 coeficientes; así, se necesitan estimar $n + pn^2$ términos. Si duda, estará sobreparametrizado en el sentido que varios de estos coeficientes podrían ser correctamente excluidos del modelo. Sin embargo, el objetivo es encontrar las interrelaciones importantes entre las variables y no realizar predicciones de corto plazo. Imponer restricciones de nulidad inadecuadas puede desperdiciar información importante. Mas aun, los regresores pueden ser altamente colineales, lo cual implica que los test *t* pueden no ser una guía confiable para descartar parámetros del modelo.

El lado derecho de (8) contiene solo variables predeterminadas y los términos de error se suponen serialmente incorrelacionados con varianza constante. Así, cada ecuación del sistema se puede estimar utilizando mínimos cuadrados ordinarios (***O***rdinary ***L***east ***S***quares – OLS). Las estimaciones mediante OLS serán eficientes y asintoticamente eficientes. Aunque los errores están correlacionados entre las ecuaciones, las regresiones SUR (***S***eemingly ***U***nrelated ***R***egressions) no agregan eficiencia al procedimiento de estimación, dado que ambas ecuaciones poseen idénticas variables en su lado derecho.

Respecto a la estacionariedad en los modelos VAR, Sims (1980) y otros, recomiendan no diferenciar aun cuando las variables contengan raíz unitaria. Ellos señalan que el objetivo del análisis VAR es determinar interrelaciones entre las variables, no la estimación de los parámetros. El argumento fundamental en contra de la diferenciación es que "se deshace" de información relevante acerca de co-movimientos en los datos (como ante la posible existencia de relaciones de cointegración). De manera similar, se discute que los datos no necesitan ser despojados de tendencia. En un modelo VAR, una variable con tendencia será bien aproximada mediante una raíz unitaria con deriva (drift). Sin embargo, la visión de la mayoría es que la forma de las variables en el VAR debe ser una mímica del verdadero proceso generador de datos. ***Esto es particularmente cierto si la idea es estimar un modelo estructural.***

3.3. Identificación.

Para ilustrar el proceso de identificación, se retorna al VAR estructural de primero orden y dos variables, representado por (1) y (2). Debido a la retroalimentación (feedback) inherente en el sistema, estas ecuaciones no pueden ser estimadas en forma directa. La razón es que z_t esta correlacionada con el termino de error ε_{yt} y y_t con el termino de error ε_{zt}. Las técnicas de estimación convencionales, requieren que los regresores estén incorrelacionados con el término de error. Al estimar el VAR en forma estándar no existe este problema. Mediante OLS se pueden estimar los dos elementos de A_0 y los cuatro

elementos de A_1. Mas aun, obteniendo los residuos de ambas regresiones es posible calcular estimaciones para las varianzas de e_{1t}, e_{2t}; y de la covarianza entre e_{1t} y e_{2t}. La cuestión es la posibilidad de recuperar toda la información presente en el sistema primitivo del sistema estimado (1) y (2). En otras palabras, la pregunta es: ¿es posible identificar la forma primitiva dadas las estimaciones OLS del modelo VAR en la forma de (4) y (5)?

La respuesta a esta pregunta es "no, a menos que estemos dispuestos a restringir de manera apropiada el sistema original". La razón es clara si se compara el número de parámetros en el VAR estructural con el número de parámetros recuperados del VAR en forma estándar. Al estimar (4) y (5) surgen seis estimaciones de coeficientes $(a_{10}, a_{20}, a_{11}, a_{12}, a_{21}, a_{22})$ y los valores calculados de $Var(e_{1t})$, $Var(e_{2t})$ y $Cov(e_{1t}, e_{2t})$. Sin embargo, el sistema primitivo (1) y (2) contiene 10 parámetros. Adicionalmente a los dos coeficientes de intercepto b_{10} y b_{20}, los cuatro coeficientes autoregresivos $\gamma_{11}, \gamma_{12}, \gamma_{21}$ y γ_{22}, y los dos coeficientes de retroalimentación b_{12} y b_{21}, habrá dos desvíos estándar, σ_y y σ_z. En total, el sistema primitivo contiene diez parámetros, mientras que el VAR estimado arroja solo nueve parámetros. A menos que se este dispuesto a restringir uno de los parámetros, no será posible identificar el sistema primitivo; las ecuaciones (1) y (2) estarán sub-identificadas. Si exactamente un parámetro del sistema primitivo es restringido, el sistema estará exactamente identificado, y si más de un parámetro es restringido, el sistema estará sobre-identificado.

Una manera de identificar el modelo es utilizar el tipo de sistema recursivo propuesto por Sims (1980). Supóngase que se esta dispuesto a imponer una restricción sobre el sistema primitivo tal que el coeficiente b_{21} sea igual a cero. Escribiendo (1) y (2) con esta restricción se obtiene:

$$y_t = b_{10} - b_{12} z_t + \gamma_{11} y_{t-1} + \gamma_{12} z_{t-1} + \varepsilon_{yt} \quad (9)$$

$$z_t = b_{20} \qquad + \gamma_{21} y_{t-1} + \gamma_{22} z_{t-1} + \varepsilon_{zt} \quad (10$$

Dada la restricción (la cual puede ser sugerida por un modelo económico particular). Está claro que z_t tiene un efecto contemporáneo sobre y_t, pero y_t afecta la secuencia de z_t con un periodo de rezago. Imponer la restricción $b_{21} = 0$ significa que B^{-1} estará dada por:

$$B = \begin{bmatrix} 1 & -b_{12} \\ 0 & 1 \end{bmatrix}$$

Ahora, premultiplicando el sistema primitivo por B^{-1}, se obtiene:

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} 1 & -b_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{10} \\ b_{20} \end{bmatrix} + \begin{bmatrix} 1 & -b_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} 1 & -b_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix}$$

o

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} b_{10} - b_{12}b_{20} \\ b_{20} \end{bmatrix} + \begin{bmatrix} \gamma_{11} - b_{12}\gamma_{21} & \gamma_{12} - b_{12}\gamma_{22} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{yt} - b_{12}\varepsilon_{zt} \\ \varepsilon_{zt} \end{bmatrix} (11)$$

Estimando el sistema mediante OLS devuelve las estimaciones teóricas de los parámetros:

$$y_t = a_{10} + a_{11}y_{t-1} + a_{12}z_{t-1} + e_{1t}$$
$$z_t = a_{20} + a_{21}y_{t-1} + a_{22}z_{t-1} + e_{2t}$$

donde:

$$a_{10} = b_{10} - b_{12}b_{20}$$
$$a_{11} = \gamma_{11} - b_{12}\gamma_{21}$$
$$a_{12} = \gamma_{12} - b_{12}\gamma_{22}$$
$$a_{20} = b_{20}$$
$$a_{21} = \gamma_{21}$$
$$a_{22} = \gamma_{22}$$

Dado que $e_{1t} = \varepsilon_{yt} - b_{12}\varepsilon_{zt}$ y $e_{2t} = \varepsilon_{zt}$, se pueden calcular los parámetros de la matriz de varianzas y covarianzas como:

$$Var(e_1) = \sigma_y^2 + b_{12}^2\sigma_z^2 (12)$$
$$Var(e_2) = \sigma_z^2 (13)$$
$$Cov(e_1, e_2) = -b_{12}\sigma_z^2 (14)$$

Así, se tienen estimaciones de nueve parámetros, $a_{10}, a_{20}, a_{11}, a_{12}, a_{21}, a_{22}$, $Var(e_{1t})$, $Var(e_{2t})$ y $Cov(e_{1t}, e_{2t})$ que pueden ser sustituidos en las nueve ecuaciones de arriba a fin de resolver en forma simultanea para $b_{10}, b_{12}, \gamma_{11}, \gamma_{12}, b_{20}, \gamma_{21}, \gamma_{22}, \sigma_y^2, \sigma_z^2$.

Las estimaciones de ε_{yt} y ε_{zt} se pueden recuperar. Los residuos de la segunda ecuación son estimaciones de la serie ε_{zt}. Combinando estas estimaciones junto con la solución de b_{12} se puede calcular la estimación de ε_{yt} utilizando la relación $e_{1t} = \varepsilon_{yt} - b_{12}\varepsilon_{zt}$.

El supuesto $b_{21} = 0$ implica que y_t no tiene efecto contemporáneo sobre z_t. En (11), la restricción se manifiesta de forma tal que ambos shocks ε_{yt} y ε_{zt} afectan el valor contemporáneo de y_t, pero solo los shocks ε_{zt} afectan el valor contemporáneo de z_t. Los valores observados de e_{2t} son completamente atribuidos a shocks puros a la serie z_t. ***La descomposición de los residuos en esta forma triangular es llamada descomposición de Choleski.***

3.4. La Función de Impulso Respuesta.

Tal como una autorregresion posee una representación en medias móviles, un vector autorregresivo puede escribirse como un vector de medias móviles (VMA – ***V***ector ***M***oving ***A***verage). La representación VMA de (3) sería $x_t = \mu + \sum_{i=0}^{\infty} A_1^i e_{t-i}$ donde $\mu = \left[\bar{y} - \bar{z}\right]$. Un aspecto esencial de la metodología de Sims (1980) es esta forma de representación, ya que permite trazar el sendero temporal de los distintos shocks sobre las variables contenidas en el sistema VAR. Con propósitos ilustrativos, se continuará analizando el modelo de primer orden y dos variables. Escribiendo (4) y (5) en notación matricial, se obtiene:

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} a_{10} \\ a_{20} \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} e_{1t} \\ e_{2t} \end{bmatrix} \quad (15)$$

Utilizando la representación VMA, se obtiene:

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} \bar{y} \\ \bar{z} \end{bmatrix} + \sum_{i=0}^{\infty} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^i \begin{bmatrix} e_{1t-i} \\ e_{2t-i} \end{bmatrix} \quad (16)$$

La ecuación (16) expresa y_t y z_t en términos de ε_{yt} y ε_{zt}. De (6) y (7), el vector de errores puede ser escrito como:

$$\begin{bmatrix} e_{1t} \\ e_{2t} \end{bmatrix} = [1/(1-b_{12}b_{21})] \begin{bmatrix} 1 & -b_{12} \\ -b_{21} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix} (17)$$

de manera que (16) y (17) se pueden combinar para formar:

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} \overline{y} \\ \overline{z} \end{bmatrix} + [1/(1-b_{12}b_{21})] \sum_{i=0}^{\infty} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^i \begin{bmatrix} 1 & -b_{12} \\ -b_{21} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_{yt} \\ \varepsilon_{zt} \end{bmatrix}$$

La notación se puede simplificar definiendo una matriz ϕ_i de 2 x 2 con elementos $\phi_{jk}(i)$:

$$\phi_i = \left[A_1^i/(1-b_{12}b_{21})\right] \begin{bmatrix} 1 & -b_{12} \\ -b_{21} & 1 \end{bmatrix}$$

Así, la representación VMA de (16) y (17) puede escribirse en términos de ε_{yt} y ε_{zt}:

$$\begin{bmatrix} y_t \\ z_t \end{bmatrix} = \begin{bmatrix} \overline{y} \\ \overline{z} \end{bmatrix} + \sum_{i=0}^{\infty} \begin{bmatrix} \phi_{11}(i) & \phi_{12}(i) \\ \phi_{21}(i) & \phi_{22}(i) \end{bmatrix} \begin{bmatrix} \varepsilon_{yt-i} \\ \varepsilon_{zt-i} \end{bmatrix}$$

o en forma más compacta: $x_t = \mu + \sum_{i=0}^{\infty} \phi_i \, \varepsilon_{t-i} (18)$.

La representación en medias móviles (VMA) es una herramienta especialmente útil para examinar la interacción entre las series y_t y z_t. Los coeficientes ϕ_i pueden utilizarse para generar los efectos de los shocks ε_{yt} y ε_{zt} sobre la totalidad de los senderos temporales de y_t y z_t. Los cuatro elementos $\phi_{jk}(0)$ son multiplicadores de impacto. Por ejemplo, el coeficiente $\phi_{12}(0)$ es el impacto instantáneo de un cambio unitario de ε_{zt} sobre y_t. De la misma manera, los elementos $\phi_{11}(1)$ y $\phi_{12}(1)$ son las respuestas de un periodo a un cambio unitario en ε_{yt-1} y ε_{zt-1} sobre y_t, respectivamente. Actualizando un periodo $\phi_{11}(1)$ y $\phi_{12}(1)$ también representan el efecto de un cambio unitario en ε_{yt} y ε_{zt} sobre y_{t+1}.

Los efectos acumulados de un impulso unitario en ε_{yt} y/o ε_{zt} se pueden obtener de la suma apropiada de los coeficientes de las funciones de impulso respuesta. Por ejemplo, luego de n periodos, el

efecto de ε_{zt} sobre el valor de y_{t+n} es $\phi_{12}(n)$. Así, después de n periodos, la suma acumulada de los efectos de ε_{zt} sobre y_t es $\sum_{i=0}^{n}\phi_{12}(i)$.

Permitiendo que en n aproxime a infinito se obtienen los multiplicadores de largo plazo. Dado que y_t y z_t se asumen estacionarias, debe cumplirse para todo j y k, que: $\sum_{i=0}^{\infty}\phi_{jk}^{2}(i)$ es finito.

Los cuatro conjuntos de coeficientes $\phi_{11}(i),\phi_{12}(i),\phi_{21}(i),\phi_{22}(i)$ son llamados funciones de impulso respuesta. Graficar las funciones de impulso respuesta es una manera practica de representar el comportamiento de y_t y z_t en respuesta a los distintos shocks. En principio, puede ser posible conocer todos los parámetros del sistema primitivo (1) y (2). Con ese conocimiento, podría ser posible trazar el sendero temporal de los efectos de shocks puros de ε_{yt} o ε_{zt}. Sin embargo, esta metodología no esta disponible para el investigador dado que un VAR estimado esta sub-identificado. El conocimiento de los distintos a_{ij} y a matriz $\sum$ de varianzas y covarianzas, no es suficiente para identificar el sistema primitivo. Por la tanto, el econometrista debe imponer una restricción adicional en el sistema VAR de dos variables a fin de poder identificar los impulsos respuesta.

Una restricción de identificación posible es usar la descomposición de Choleski. Por ejemplo, es posible restringir el sistema de manera que los valores contemporáneos de y_t no tengan efectos contemporáneos sobre z_t. Formalmente, estas restricciones representadas imponiendo $b_{21}=0$ en el sistema primitivo. En términos de (17), el término de error puede descomponerse de la siguiente manera:

$$e_{1t}=\varepsilon_{yt}-b_{12}\varepsilon_{zt} \quad (19)$$
$$e_{2t}=\varepsilon_{zt} \quad (20)$$

Así, utilizando (20), todos los errores observados de e_{2t} se atribuyen a shocks de ε_{zt}. Dada la serie calculada de ε_{zt}, el conocimiento de los valores de e_{1t} y coeficientes de correlación entre e_{1t} y e_{2t}, permiten el calculo de ε_{yt} utilizando (19). A pesar que esta descomposición de Choleski restringe el sistema tal que un shock de ε_{yt} no tiene efecto directo en z_t, existe un efecto indirecto ya que valores rezagados de y_t afectan los valores contemporáneos de z_t. El punto clave es que la descomposición fuerza una asimetría potencialmente importante en el sistema dado que shocks de ε_{zt} tienen efectos

contemporáneos tanto sobre y_t como sobre z_t. Por esta razón, (19) y (20) se dice que implican un ordenamiento en las variables. Un shock de ε_{zt} afecta en forma directa a e_{1t} y a e_{2t}, pero un shock de ε_{yt} no afecta a e_{2t}. En este sentido, z_t es "anterior" a y_t.

En conclusión, la identificación requiere imponer alguna estructura en el sistema. La descomposición de Choleski provee un conjunto mínimo de supuestos que pueden ser utilizados para identificar el modelo primitivo. Es crucial comprender que la importancia del ordenamiento depende de la magnitud del coeficiente de correlación entre e_{1t} y e_{2t}.

3.5. Aplicación.

La base de datos sobre la cual se llevó a cabo el análisis empírico, corresponde a una compilación realizada a partir de información sobre market share de webservers, relevada y publicada mensualmente, en el sitio www.securityspace.com (ver anexo iii para mayor detalle).

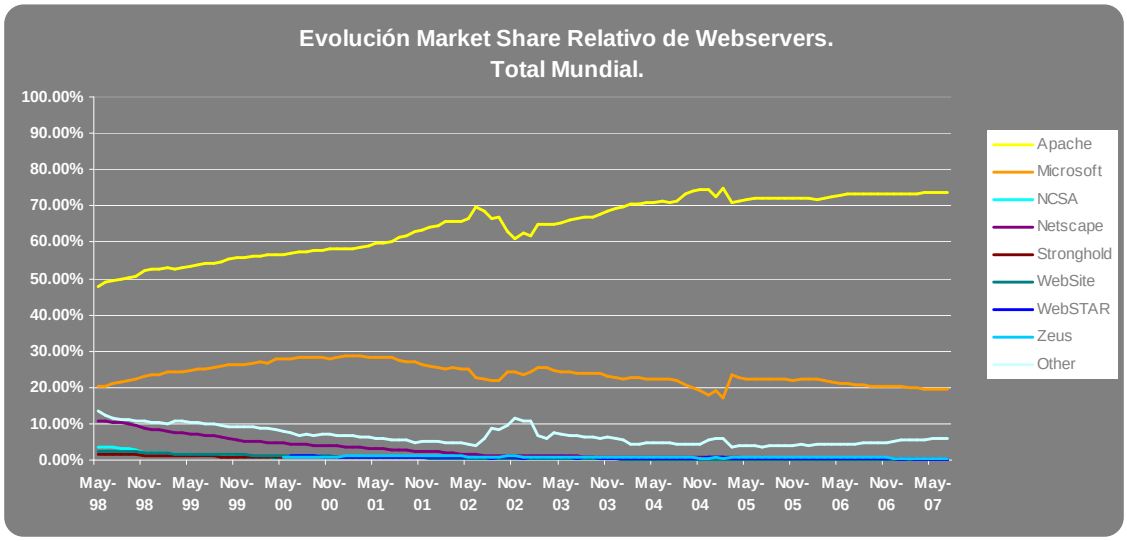
Como se refiriera previamente, la periodicidad de esta información es mensual, y el intervalo de análisis, fue desde mayo de 1998, hasta julio de 2007.

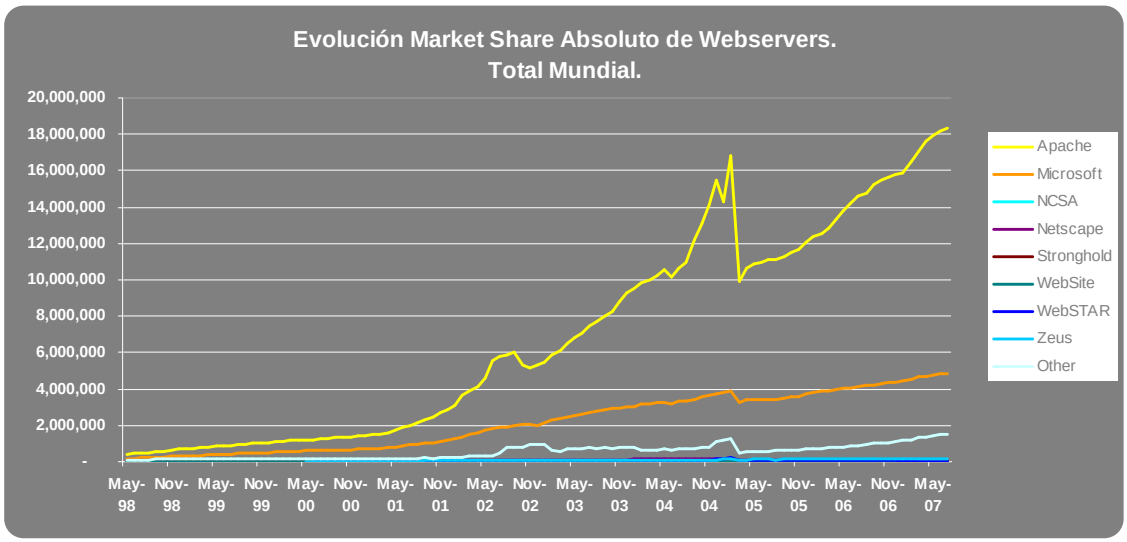
Aplicando la metodología econométrica VAR explicitada en esta sección, se estudiarán las funciones de impulso – respuesta. La existencia de externalidades de red, implicaría que un aumento del market share de uno de los competidos analizados (Apache / Microsoft), tendría el efecto de propagar durante una cierta cantidad de periodos, la captación de market share por parte este competidor. Del mismo modo, la externalidad de red se evidenciaría también, si ante un incremento en el market share de uno de los competidores, su rival perdiera mercado o ganara muy poco (o nada), durante un numero de periodos posteriores.

Se obtuvieron los resultados a continuación detallados para la series correspondientes al mercado mundial en su conjunto, y la Argentina en forma separada.

3.5.1. Mercado Mundial.

Los siguientes gráficos, muestran la evolución del market share para los principales competidores del mercado de webservers a nivel mundial; tanto en valores absolutos como relativos.





El market share relativo, muestra al inicio, un comportamiento de captación por parte de los principales competidores (Apache y Microsoft), a expensas de los competidores menores. A partir de (aproximadamente) Mayo 2001, el crecimiento en la presencia de mercado de Apache y Microsoft, comenzó a darse mediante un proceso de canibalización entre ellos mismos. El segmento de mercado abarcado por la categorización 'Otros' (Others), correspondería a productos de nicho.

La modelización VAR, se realizó sobre las series en términos relativos. Se extrajo la tendencia determinística mediante un regresor lineal y otro cuadrático (adicionalmente al término de intercepto); no

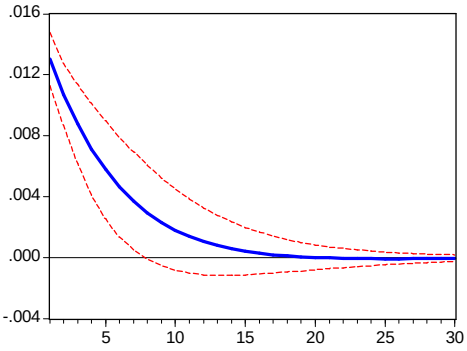
registrándose tendencia estocástica. La series resultantes, son DTR_APACHE (Apache) y DTR_MSFT (Microsoft). La especificación VAR (1, 1) resultó la de mejor ajuste.

Vector Autoregression Estimates
Date: 08/20/07 Time: 17:22
Sample(adjusted): 1998:06 2007:07
Included observations: 110 after adjusting endpoints
Standard errors in ( ) & t-statistics in [ ]

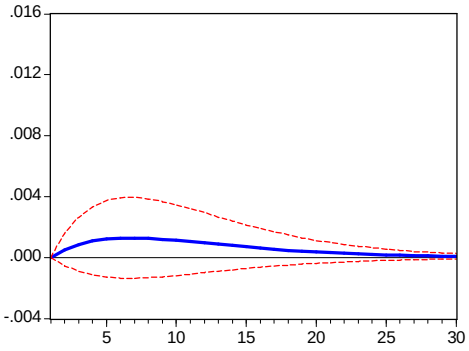
	DTR_APACHE	DTR_MSFT
DTR_APACHE(-1)	0.853119	-0.168584
	(0.06113)	(0.17701)
	[13.9555]	[-0.95239]
DTR_MSFT(-1)	0.017213	0.833420
	(0.01796)	(0.05200)
	[0.95844]	[16.0259]
C	0.000188	0.001899
	(0.00124)	(0.00360)
	[0.15146]	[0.52819]
R-squared	0.688829	0.767207
Adj. R-squared	0.683013	0.762856
Sum sq. resids	0.018153	0.152201
S.E. equation	0.013025	0.037715
F-statistic	118.4312	176.3177
Log likelihood	322.9344	205.9835
Akaike AIC	-5.816989	-3.690610
Schwarz SC	-5.743340	-3.616960
Mean dependent	0.000193	0.001552
S.D. dependent	0.023134	0.077448
Determinant Residual Covariance		1.47E-07
Log Likelihood (d.f. adjusted)		553.2676
Akaike Information Criteria		-9.950320
Schwarz Criteria		-9.803021

Respuesta a un desvío estandar de Cholesky. Innovaciones ± 2 errores estandar.

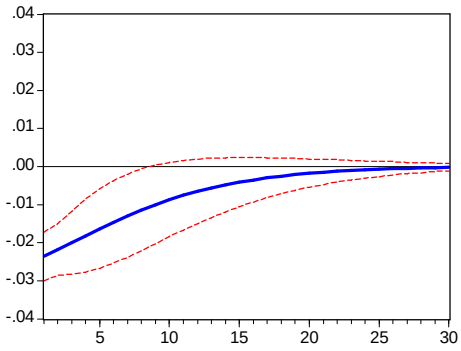
Resuesta de DTR_APACHE a DTR_APACHE



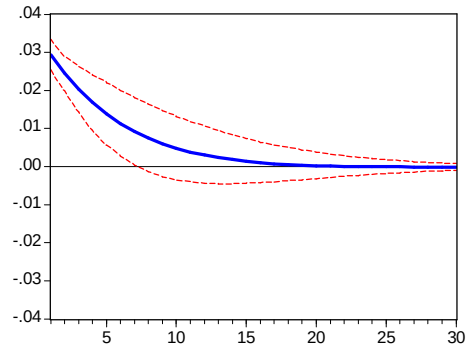
Resuesta de DTR_APACHE a DTR_MSFT



Resuesta de DTR_MSFT a DTR_APACHE

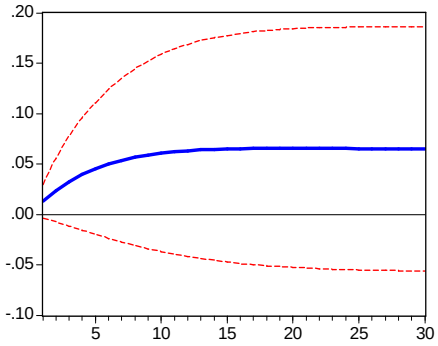


Resuesta de DTR_MSFT a DTR_MSFT

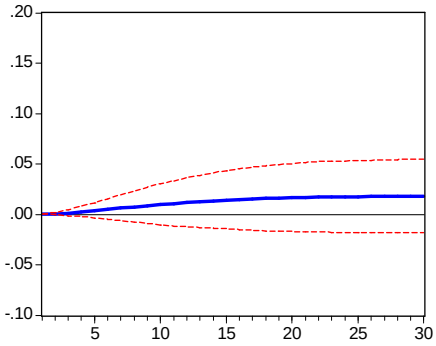


Respuesta acumulada a un desvío estandar de Cholesky. Innovaciones ± 2 errores estandar.

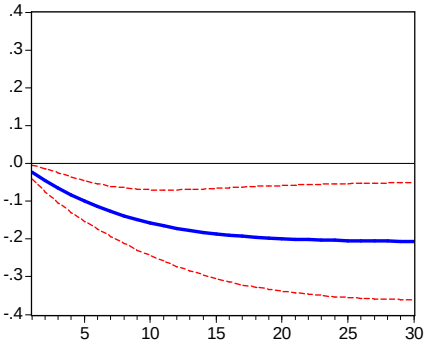
Respusta de DTR_APACHE a DTR_APACHE



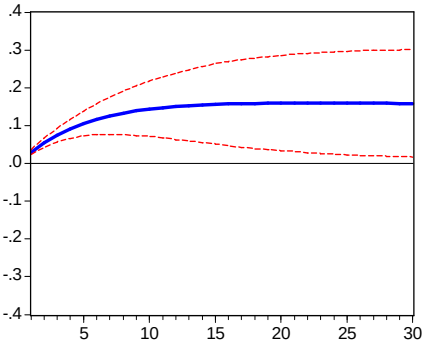
Respusta de DTR_APACHE a DTR_MSFT



Respusta de DTR_MSFT a DTR_APACHE

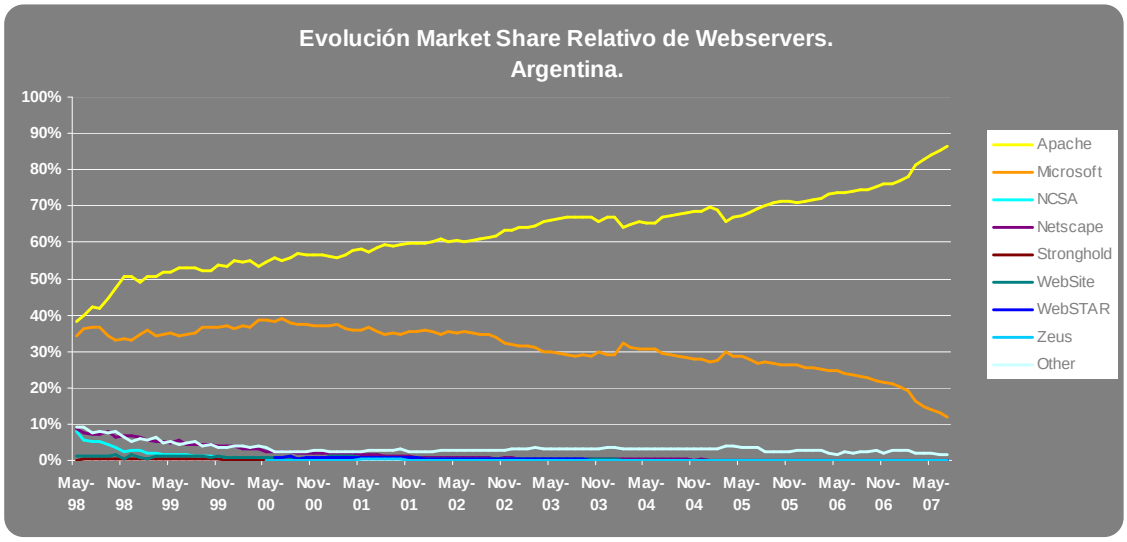


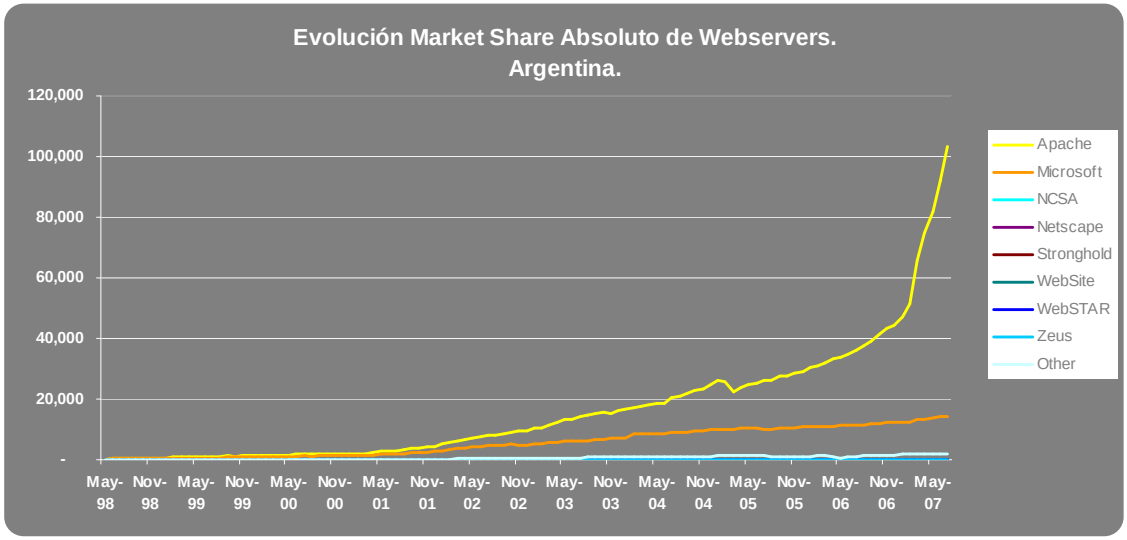
Respusta de DTR_MSFT a DTR_MSFT



3.5.2. Mercado Argentina.

A nivel del mercado argentino, se muestran a continuación, los gráficos de evolución análogos a los de la sub-sección precedente.





Al igual que en las series correspondientes al mercado global, el market share relativo, muestra al inicio, un comportamiento de captación por parte de los principales competidores (Apache y Microsoft), a expensas de los competidores menores; mientas que posteriormente, se daría una canibalización entre Apache y Microsoft en busca de mayor presencia de mercado. El segmento correspondiente a 'Otros', se compondría de productos de nicho.

También para esta base de datos, se llevo a cabo la modelización VAR sobre las series en términos relativos. La extracción de tendencia determinística, se realizó mediante regresores: lineal, cuadrático y exponencial (adicionalmente al término de intercepto); no registrándose tendencia estocástica. La series

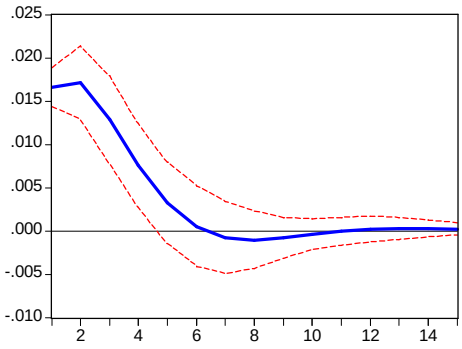
resultantes, son DTR_APACHE (Apache) y DTR_MSFT (Microsoft). La especificación VAR (1, 2) resultó ser la de mejor ajuste.

Vector Autoregression Estimates
Date: 08/19/07 Time: 14:00
Sample(adjusted): 1998:07 2007:07
Included observations: 109 after adjusting endpoints
Standard errors in ( ) & t-statistics in [ ]

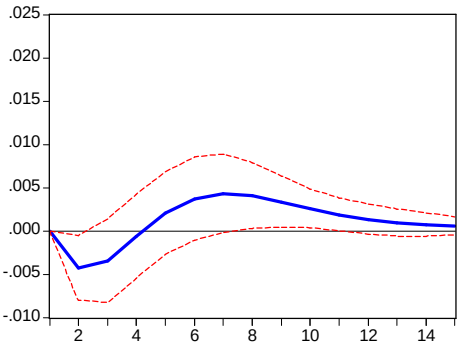
	DTR_APACHE	DTR_MSFT
DTR_APACHE(-1)	0.739786	0.215401
	(0.14631)	(0.30779)
	[5.05635]	[0.69984]
DTR_APACHE(-2)	0.031873	-0.118810
	(0.12444)	(0.26177)
	[0.25614]	[-0.45387]
DTR_MSFT(-1)	-0.184426	1.293159
	(0.07915)	(0.16652)
	[-2.32994]	[7.76596]
DTR_MSFT(-2)	0.226319	-0.523674
	(0.07668)	(0.16131)
	[2.95143]	[-3.24633]
C	0.001262	0.000592
	(0.00161)	(0.00339)
	[0.78369]	[0.17458]
R-squared	0.748187	0.688189
Adj. R-squared	0.738502	0.676196
Sum sq. resids	0.028718	0.127089
S.E. equation	0.016617	0.034957
F-statistic	77.25140	57.38380
Log likelihood	294.5027	213.4406
Akaike AIC	-5.311976	-3.824597
Schwarz SC	-5.188519	-3.701141
Mean dependent	0.003068	-0.000583
S.D. dependent	0.032495	0.061432
Determinant Residual Covariance		1.49E-07
Log Likelihood (d.f. adjusted)		547.3744
Akaike Information Criteria		-9.860080
Schwarz Criteria		-9.613167

Respuesta a un desvío estandar de Cholesky. Innovaciones ± 2 errores estandar.

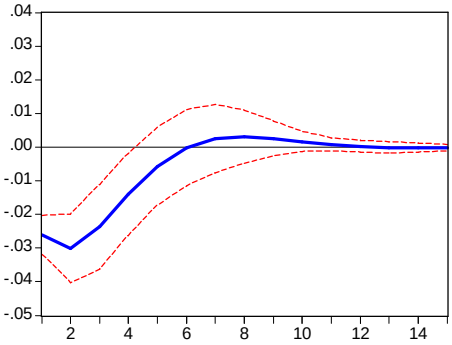
Respuesta de DTR_APACHE a DTR_APACHE



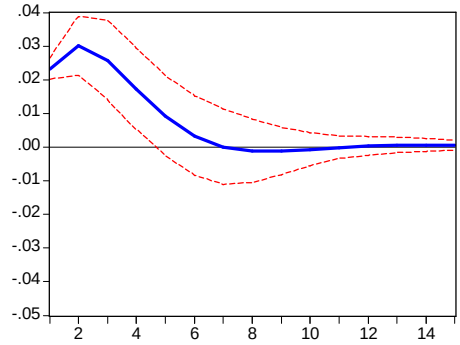
Respuesta de DTR_APACHE a DTR_MSFT



Respuesta de DTR_MSFT a DTR_APACHE

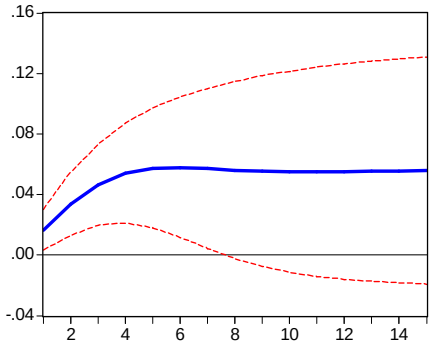


Respuesta de DTR_MSFT a DTR_MSFT

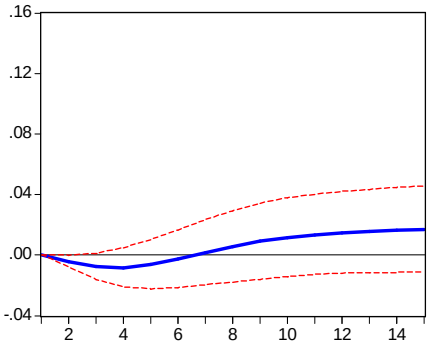


Respuesta acumulada a un desvío estandar de Cholesky. Innovaciones ± 2 errores estandar.

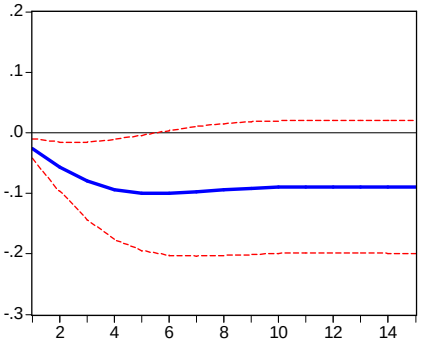
Respuesta de DTR_APACHE a DTR_APACHE



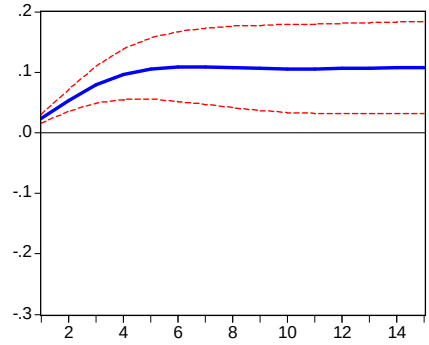
Respuesta de DTR_APACHE a DTR_MSFT



Respuesta de DTR_MSFT a DTR_APACHE



Respuesta de DTR_MSFT a DTR_MSFT



4. ITERPRETACIÓN DE LOS RESULTADOS, CONCLUSIONES E IMPLICANCIAS.

Las funciones de impulso – respuesta para market shares de Apache y Microsoft en el mercado internacional, muestran que un shock positivo para Apache, propaga la captación de mercado durante aproximadamente 15 periodos posteriores. El efecto sobre el market share de Microsoft, sería negativo, agotándose en aproximadamente 20 periodos.

Un shock positivo para Microsoft, aumenta su market share durante aproximadamente 15 periodos (pero con mayor intensidad absoluta que la observada en la situación análoga para Apache). No obstante, el efecto observado de este shock sobre la fracción de mercado ocupada por Apache en los periodos sucesivos, no es decremental. Si no que solo logra (prácticamente) bloquear su crecimiento.

Lo expresado en los párrafos anteriores, respaldaría la existencia de externalidades de red en este mercado: un crecimiento de la base instalada, guía a un crecimiento inercial de la fracción de mercado en los periodos inmediatos posteriores para el competidor que recibe el impulso favorable; implicando un efecto de decremento o bloqueo de crecimiento, sobre el otro oferente.

Al realizar este análisis sobre los resultados observados para el mercado argentino, se llega a conclusiones cualitativamente similares a las obtenidas para el mercado mundial, si bien existirían diferencias cuantitativas: un shock positivo para Apache, se expandiría durante 6 periodos, con un efecto negativo sobre el market share de Microsoft durante un idéntico intervalo de tiempo; mientras que un shock positivo para Microsoft, propagaría su efecto por 7 periodos, produciendo un leve decremento sobre la fracción de mercado de Apache, que sería neutralizado posteriormente.

Podría entonces concluirse, que existirían externalidades de red tanto para el mercado internacional, como el local.

Como se refiriera previamente, la verificación de externalidades de red entraña dos aspectos fundamentales:

- Que se dará una sub – optimalidad si se favorece (aún con la no – intervención) la coexistencia de más de un estándar.
- Aún cuando el estándar de interfases cerradas y propietarias sea cualitativamente inferior, continuará manteniendo su predominancia a menos que el estado actúe coordinando la formación de base instalada de un estándar alternativo, tal como el planteado por el OSS (efectos *lock – in* y *path dependence*).

Así, las acciones orientadas a implantar un nuevo estándar serán efectivas si son "drásticas": cuando el gobierno es capaz de influir sobre una gran masa de consumidores entonces el mercado se inclina hacia el software open source y las externalidades de red alcanzan su máximo incrementando el bienestar.

Una vez analizada la posibilidad de obtener no solo un no-decremento, si no incluso un incremento en el bienestar agregado mediante la adopción de software FLOSS, existiría respaldo teórico para avanzar sobre las fronteras abiertas por este modelo productivo, y sus ventajas de largo plazo asociadas.

El OSS brindaría a los países en desarrollo una posibilidad de incentivar la expansión de sus sectores de IT: uno de los componentes más importantes del costo total de propiedad del software (TCO) es el costo laboral de los administradores de sistemas y personal de soporte técnico. Estos costos pueden ser mucho mas bajo en países con menores costos laborales (países en desarrollo). Dado que el precio de compra del software open source es sustancialmente mas bajo que las alternativas propietarias, el TCO – el cual incluye tanto al costo del software como al costo laboral necesario para mantenerlo – podría ser significativamente mas bajo en estos países. Esto podría incentivar a las empresas transnacionales a iniciar sus procesos de migración hacia entornos abiertos (OSS), en países en desarrollo con mano de obra capacitada en este tipo de software.

Para ello, serían importantes las políticas de promoción de OSS por parte del gobierno, (mediante las vías antes recomendadas) que contribuyan a la formación de profesionales de la industria informática con dominio de este tipo de software. En este sentido, una ventaja que presenta el OSS es la posibilidad para los estudiantes de acceder a los códigos fuente del mundo real, facilitando su inserción laboral en entornos OSS.

En línea con lo anterior, la difusión de entornos con interfases abiertas alentaría el desarrollo de software comercial compatible, aumentando las posibilidades de crecimiento para la industria del software local.

Para una presentación detallada de los potenciales beneficios sobre el desarrollo de sectores de IT local en países periféricos, asociados al apoyo oficial al movimiento FLOSS, como así también de las principales propuestas de implementación; véase el trabajo de Sanjiwa Weerawarana y Jivaka Weeratunga, Open Source in Developing Countries (Weerawarana & Weeratunga, 2004).

Resulta importante señalar que existirían costos significativos asociados a las medidas adoptadas tendientes a apoyar la creación de base instalada para el movimiento FLOSS, fundamentalmente referidos a la migración de estándares.

Un análisis que excede el alcance de la presente investigación, debiera centrarse en explicar porque algunos mercados de software presentan concentración (duopolio / monopolio) mientras otros exhiben diversificación (ejemplo: webservers vs. servidores de correo). Una posible explicación sería la intensidad del efecto de red. Futuras investigaciones, podrían orientarse en esta línea.

Desde el punto de vista metodológico, el presente trabajo aporta una vía alternativa para el análisis empírico de externalidades de red.

Enfoques tradicionales, tales como el de Brynjolfsson y Kemerer (Brynjolfsson y Kemerer, 1995), se basan en la estimación de modelos hedónicos, a fin de detectar la existencia (y magnitud) de las externalidades de red, mediante la relación observada en las ecuaciones de precio, entre base instalada y precio del software (controlando por otros factores).

A diferencia de esta propuesta metodológica, el análisis desarrollado en el presente estudio, se basó en la metodología econométrica VAR. La existencia de externalidades de red, fue así determinada, en función del impacto futuro que tendría sobre la propia base instalada y la del competidor más cercano, un incremento de la base instalada propia en el período corriente. Bajo este enfoque metodológico, se evita enfrentar ciertas limitaciones y distorsiones, presentes en las ecuaciones de precio, a la hora de estimar efectos de red.

5. BIBLIOGRAFÍA.

- Arthur W. B. Competing Technologies, Increasing Returns, and Lock-in by Historical Events. The Economic Journal 99 (1989), pages 116-131.
- Arthur W. B. Increasing Returns and Path Dependence in the Economy. University of Michigan Press, Ann Arbor. 1994.
- Arthur W. B. Increasing Returns and the New Word of Business. Harvard Business Review, pages 100-109. 1996.
- Arthur W. B., Ermeliov Y., Kaniovski Y. On Generalised Urn Schemes of Polya Kind. Cybernetics 19, pages. 61-71. 1983.
- Bonaccorsi, A. & Rossi, C. Why Open Source con succed. LEM Working Paper Series. July 2002.
- Brynjolfsson, Erik & Kemerer, Chris F. "Network Externalities in Microcomputer Software: An Econometric Analysis of the Spreadsheet Market". Management Science. July 1995
- Casadesus - Masanell, R. and Ghemawat, P. (2003). Linux vs. Windows: Modeling Competition between Open-Source and Closed Software. Harvard Business School, mimeo.
- Church, Jeffrey & Gandal, Neil. Network effects, software provision and standardization. The Journal of Industrial Economics, Volume XL (March 1992), pp. 85-103.
- Comino, Stefano & Manenti, Fabio. Open Source vs. Closed Source Software: Public Policies in the Software Market. June 2003.
- Dalle, Jean-Michel & Jullien, Nicolas. 'Libre' software: turning fads into institutions? Noviembre 2001.
- David P. CLIO and the Economics QWERTY. American Economic Review 75 (1985), pages 332-337.
- Ducheneaut, Nicolas. Socialization in an Open Source Software Community: A Socio-Technical Analysis. 2005.
- Enders, W. Applied Econometric Time Series. John Wiley and Sons. 1995.
- Evans, D. S. (2002). Politics and Programming: Government Promoting Open Source Software. in Hahn (2002).
- Evans, D. S. and Reddy, B. (2002). Government Preferences for Promoting Open-Source Software: a Solution in Search of a Problem. N.E.R.A. working paper.
- Favero, C. A. Applied Macroeconometrics. Oxford University Press. 2001.

- Gandal Neil. Competing Compatibility Standards and Network Externalities in the PC Software Market. The Review of Economics Statistics, Vol. 77, No. 4 (Nov., 1995), pp. 599-608.
- Ghosh, Rishab Aiyer & Schmidt, Philipp. Open Source and Open Standards: A New Frontier for Economic Developmente? United Nations University - Policy Brief. Number I, 2006.
- Greene, W. H. Econometric Analysis. Prentice Hall. 1980
- Hahn Jong-Hee. The welfare efect of quality degradation in the presence of network externalities. Junio 2003.
- Hertel Guido, Niedner Sven & Herrmann Stefanie. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. 2003.
- Johnson, Justin P. Economics of Open Source, mimeo, Cornell University. 2001
- Kasper Edwards. An economic perspective on software licenses—open source, maintainers and user-developers. June 2004.
- Katz Michael & Saphiro Karl. Network Externalities, Competition and Compatibility. The American Economic Review, Vol. 75, No. 3 (Jun., 1985), pp. 424 - 440.
- Lakhani Karim & Hippel Eric Von. How open source software works: “free” user-to-user assistance. Julio 2002.
- Lerner Josh & Tirole Jean. The Simple Economics of Open Source.
- Lopez Sanchez, J. I & Arroyo Barrigüete, J.L. Externalidades de red en la economía digital: una revisión teórica. Departamento de Organización de Empresas. Universidad Complutense de Madrid. 2004.
- Schelling, T. Micromotives and macro beavhior. New York, Norton. (1978).
- Schmidt, Klaus & Schnitzer, Monika. Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market. November 2002.
- Sims, Ch. Are forecasting models usable for policy analysis?. Federal Reserve Bank of Minneapolis Quarterly Review. Winter 1986.
- Sims, Ch. Macroeconomics and reality. Econometrica, Nro. 48. 1980.

- Välimaki, Mikko & Oksanen Ville. The impact of free and open source licensing on operating system software markets. June 2004.
- Varian, Hal R. & Shapiro, Carl. Linux Adoption in the Public Sector: An Economic Análisis. Dec. 2003.
- Von Krogh, Georg & Von Hippel, Eric. Special issue on open source software development. 2003.
- Von Krogh, Georg; Spaeth, Sebastian & Lakhani, Karim. Community, joining, and specialization in open source software innovation: a case study. 2003.
- Weber Steven. The Political Economy of Open Source Software. BRIE Working Paper 140. E-conomy ProjectTM. Working Paper 15. June 2000.
- Weerawarana, S & Weeratunga, J. Open Source in Developing Countries. Swedish Internarional Development Cooperation Agency – SIDA. January 2004.
- West Joel & Gallagher Scott. Challenges of open innovation: theparadox of firm investment in open-source software.
- Wet Joel. How open is open enough? Melding proprietary and open source platform strategies. 2003.
- Witt, U. Lock-in vs. Critical Masses. Industrial Changes under Network Externalities. International Journal of Industrial Organisation 15 (1997), pages 753-772.
- Witt, U. Path-dependence in Institutional Change. In: K. Dopfer (ed.), The Evolutionary Principles of Economics. Cambridge, Cambridge University Press. (2000).

6. ANEXOS.

1. Series temporales evolución market shares. Total mundial.


base_acumulada_mundo.xls

2. Series temporales evolución market shares. Argentina.


base_acumulada_argentina.xls

3. Metodología y clasificación empleada por Security Space.

Security Space, recolecta información publica. Por ejemplo, recopila paginas web, y graba las respuestas devueltas por los servidores, del mismo modo que cualquier requerimiento que un usuario podría realizar a un servidor. Las series presentadas, comienzan en Mayo 1998, con periodicidad mensual. En algunos casos, los 'crawlers', visitan un subconjunto de la web. En otros, todos los servidores de los que se tiene conocimiento, dependiendo de la información que se busca recolectar.

Al visitar mensualmente los webservers, se envía un requerimiento http con el siguiente aspecto (ejemplo):

```
HEAD / HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; SecuritySpace WebSurvey;
http://www.securityspace.com )
Accept: text/plain,text/html
```

Brevemente, este requerimiento se dirige a la home page del sitio web en cuestión. Pero, a fin de optimizar el ancho de banda, se requiere solo el componente del encabezado HTTP de la pagina web. De aquí, el uso de 'HEAD', en lugar de 'GET' (que sería lo utilizado por un usuario normal). Esto ahorra ambas cosas: el ancho de banda de Security Space y el de los sitios visitados. En algunos casos, el web server tratará un requerimiento de HEAD como un GET, retornando la pagina web completa. En estos casos, se descarta la pagina web retornada, reteniendo solo el encabezado (header) HTTP. La cadena

Usuario-Agente para la mayoría de los usuarios identifica el tipo de browser que están utilizando. En el caso de Security Space, se identifica su crawler con esta cadena (string).

Se visitan los sitios considerados bien-conocidos, definidos como sitios que tiene vínculos que apuntan a el de (al menos) otros sitio categorizado como bien-conocido. Así, al visitar un sitio, es porque se sabe de él a través de un vinculo desde otro sitio.

Si un sitio deja de responder a los requerimientos enviados durante 3 meses consecutivos, automáticamente es removido de la encuesta. De este modo, la lista de servidores conocidos, permanece al día.

Debido a la aplicación de esta técnica, se visitan solo el 10% de los sitio en la web; ya que aproximadamente el 90% de los sitios web serían 'franja' (fringe), tales como squatters de dominio, paginas web personales, etc. , todos los cuales se consideran como no importantes para el resto de la comunidad web (porque ningún otro los considera los suficientemente importante como para vincularlos).

Se corre un crawler general que se comporta como cualquier otro, visitando sitios web correspondientes a un subconjunto de la web cada mes. Este crawler descarga paginas completas, y toma nota de varios atributos de estas paginas (tales como la localización de objetos embebidos como imágenes, marcos, etc.).

El objetivo de este crawler es descubrir vínculos (links) a nuevos sitios de los que aun no se tenia conocimiento, como así también para tomar nota de varios atributos en uso en sitios web.

Las visitas a sitios web, se realiza de manera aleatoria. Se toma la lista completa de sitios conocidos, aleatorizándose el orden, y luego se comienza a visitar los sitios, uno tras otro. El crawler es estructurado para penetrar la lista completa una vez por año, lo cual significa, que nunca se realiza una 'araña' (spider) a cada sitio más de una vez al año.

Al realizarse los reportes y gráficos, se agrupan distintos tipos de servidores correspondientes a proveedores particulares. A continuación, se muestran las agrupaciones realizadas:

- **Netscape** es la suma de los siguientes servidores: Netscape-Enterprise, Netscape-Commerce, Netscape-Communications, Netscape-FastTrack, Netscape-Catalog, y Netscape-Administrator.
- **Microsoft** es la suma de los siguientes servidores: Microsoft-IIS, Microsoft-PWS, Microsoft-PWS-95, Microsoft-ELG, y Microsoft-Internet-Information-Server.
- **WebSite** es la suma de WebSite (por O'Reilly) y WebSitePro.